Automation in Construction 180 (2025) 106552

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

$Vault - Z$: Automated system for free-form modular pavilion design

Machi Zawidzki [a,b,*], Jacek Szklarski [b]

[a] Łukasiewicz Research Network - Industrial Research Institute for Automation and Measurements PIAP, Poland
[b] Institute of Fundamental Technological Research, Polish Academy of Sciences, Poland

ARTICLE INFO

Keywords:
Vault-Z
Extremely modular system
Dome
Pavilion
Graph evolution algorithm

ABSTRACT

This paper introduces an innovative system for the automated creation of free-form modular pavilions. A graph-theoretic Evolutionary Algorithm is implemented to generate floor plans composed of only one basic module. The process is formulated as a constrained multi-objective optimization, where the pavilion structure must be sound, the number of modules and pavilion dimensions are specified, and additional objectives, such as floor-plan roundness, site shape, and obstacle avoidance, are considered. The effectiveness of the proposed approach is demonstrated through examples capturing various scenarios: enforcing circular shapes, maximizing enclosed areas (courtyard size), and avoiding obstacles while maximizing covered areas. $Vault - Z$ is particularly suitable for special types of construction contexts, including temporary structures, post-disaster settlements, and extreme environment outposts. This paper aims to inspire further research on optimization methods for construction systems that enable rapid deployment, reconfiguration, re-use and disassembly.

1. Introduction

Domes of all sizes and purposes have been built for thousands of years. Widely regarded as the earliest known examples of dome-shaped structures, the mammoth-bone huts discovered in Ukraine in the 1960s are believed to be between 15,000–25,000 years old. Dome roofs were traditionally used in extremely hot and dry climates [1], as the dome shape reduces solar heat gains by limiting the surface area exposed to direct solar radiation [2]. The experimental results [3] from Harran, a hot arid region of Anatolia, Turkey, showed that interior conditions were relatively comfortable even under extreme outside conditions during the summer. This was attributed to the dome shape, the thermal mass of the construction materials, and natural ventilation. The thermal performance of domes in cold climates is also advantageous. For example, simulations of the energy performance of the ensemble dome-house located in Yellowknife, Canada (at 61°N latitude) showed substantial energy consumption savings of approximately 19% compared with an isolated house.

The structural advantage of dome-like structures lies in the fact that, in the material forming the arc in cross-section, one type of stress – namely compression – dominates. As a result, the amount of traditional reinforcement can be significantly reduced, with most reinforcement being auxiliary and required by building code requirements. Therefore, it is possible to construct domes using masonry with stone or brick elements with little or no formwork. It is also possible to create sprayed structures using unconventional reusable formwork, e.g., pneumatic. Dome-like structures are currently envisioned for lunar and Martian bases [4].

Beyond the functional challenges of circular plans and furniture placement along arched walls, acoustics are also a major disadvantage of domed buildings. Spaces surrounded by smooth concave surfaces are often considered unsuitable for the effective presentation of speeches or musical performances because the reflected sound from concave surfaces does not diffuse properly within a room and may create sound focal points and dead spots under certain geometric conditions [5]. Practical guidelines for the proper acoustic design of circular rooms and domes have been proposed in [5,6].

The number of completed architectural domes is relatively small; however, in the 20th century, a few residential dome dwellings, usually experimental, were commissioned.

Kopulaki, Poland

In 1961–1966, an experimental project, commonly called *Kopulaki*, was conducted in Warsaw, Poland (Fig. 1).

Each house consisted of three connected domes (compare with Fig. 13.2). The construction turned out to be too costly, and only eight of the planned seventy dome houses were completed.

Bolwoningen — the "ball house" project, the netherlands

Bolwoningen is a residential project of 50 units in Den Bosch, the

* Corresponding author at: Łukasiewicz Research Network - Industrial Research Institute for Automation and Measurements PIAP, Poland.
E-mail address: zawidzki@piap.lukasiewicz.gov.pl (M. Zawidzki).

https://doi.org/10.1016/j.autcon.2025.106552
Received 25 January 2025; Received in revised form 13 September 2025; Accepted 18 September 2025
Available online 1 October 2025

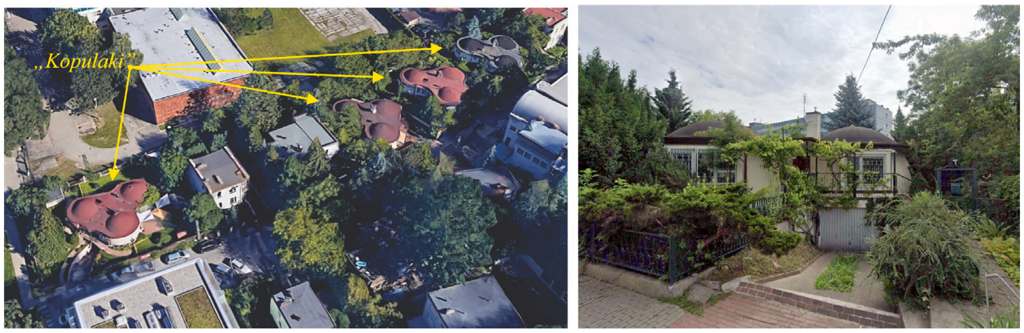


Fig. 1. On the left: the aerial view showing four unremodeled distinctive triple-dome structures. On the right: the street view showing one of the houses (Google Maps, 2024 [7]).

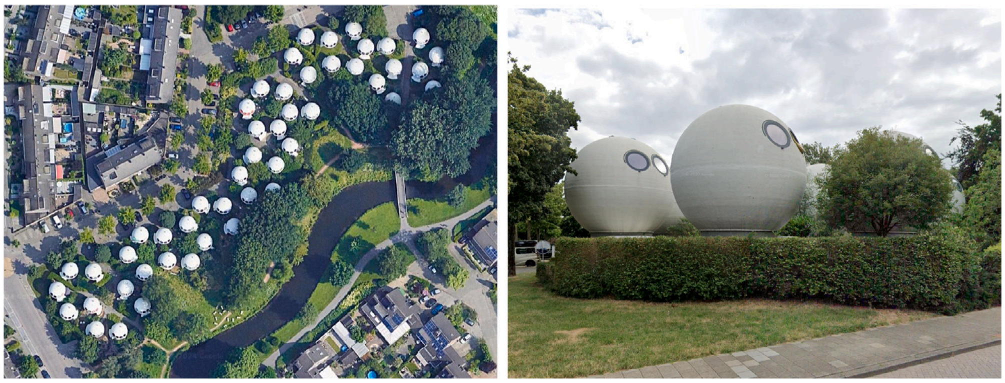


Fig. 2. *Bolwoningen* (1984–1990), Den Bosch, the Netherlands. The aerial [8] and street [9] views (Google Maps).

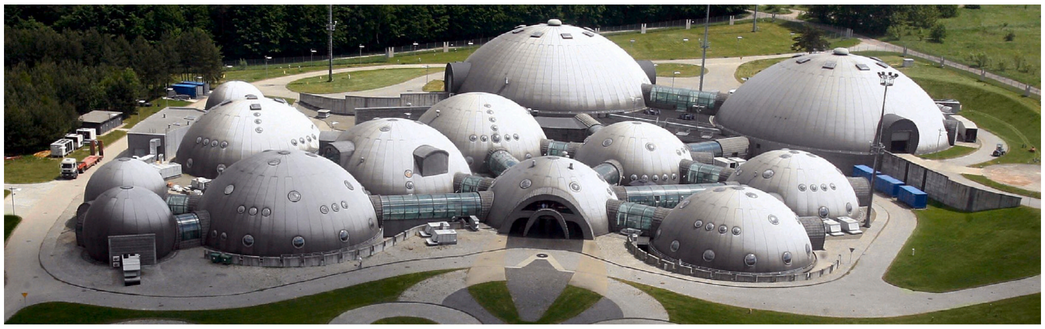


Fig. 3. Aerial view of Alvernia Studios — the largest and most modern film studio in Poland. Photograph ©Alvernia Studios.

Netherlands. It aimed to challenge conventional architectural norms and introduce a futuristic and unconventional living space. The houses were installed from 1984 to 1990 and have been continuously occupied ever since (Fig. 2).

The prefabricated fiberglass-reinforced polyester shells could be erected within one day. This type of lightweight structure (1250 kg) does not require a permanent foundation or extensive maintenance, and consumes little energy. Each house's total floor space is 55 m^2, with a diameter of 5.5 m.

Alvernia studios complex, Poland

A relatively recent example is Alvernia Studios complex, built from 2000 to 2002 in Nieporaz, Poland. The reinforced concrete domes covered with metallic cladding (with diameters at the base of: 50, 30, 25 and 12 m, and heights of: 15.90, 12.10, 9.95 and 9.80 m, respectively) are connected by concrete and glass corridors, as shown in Fig. 3.

The introduction above (Section 1) outlines the architectural context of dome-like constructions. Examples of historical structures from different periods and at different scales were shown. The main advantages and disadvantages of architectural structures with curved walls were indicated.

Section 2 of this article describes the concept of an extremely modular $Vault-Z$ system for creating pavilions of any shape based on only one type of module. This section shows the geometric properties of the module, illustrated with examples demonstrating the capabilities of the system. These examples include emulations of historical cases with $Vault-Z$. Next, preliminary work on a physical prototype of a simple dome-shaped pavilion consisting of 12 modules is presented. At the end of this section, a mathematical description of the creation of $Vault-Z$ pavilion floor-plans formulated as an optimization problem is presented. Section 3 provides a detailed description of the graph-based optimization framework, including the fitness function, and the evolutionary algorithm with its components (mutation, crossover and selection). Evolutionary operations are illustrated with basic examples.

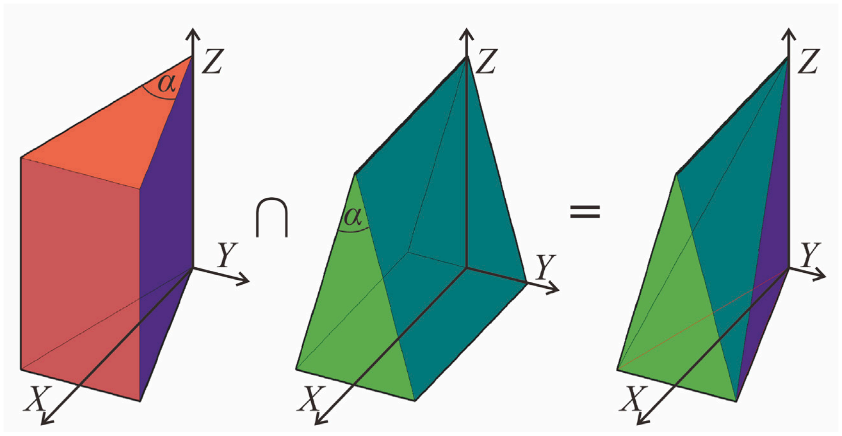


Fig. 4. Construction of the allowable solid (**AS**). Angle α is the same in planes XY and YZ, and depends on the desired number of segments in the circular approximation. In this case, it is 12, thus $\alpha = \frac{\pi}{6}$.

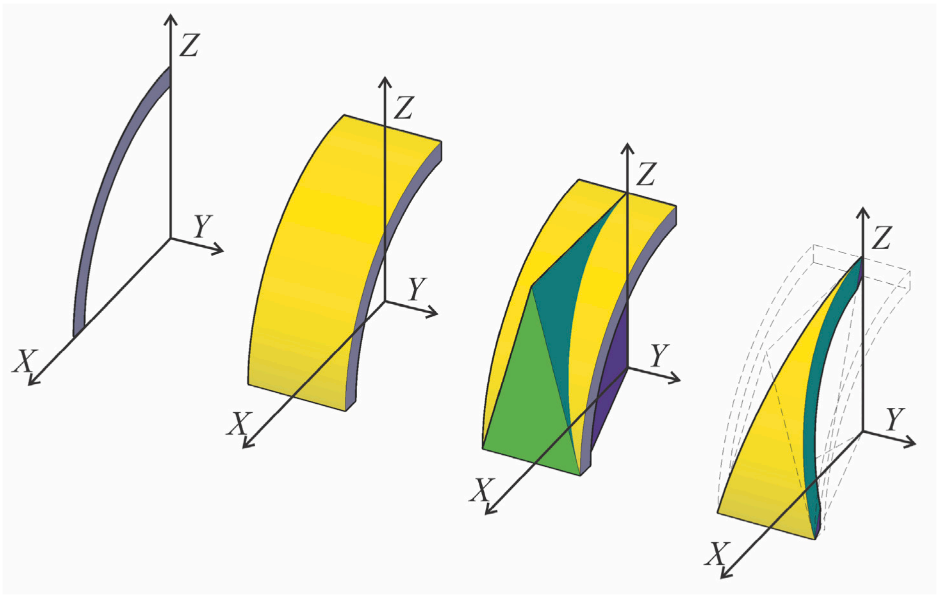


Fig. 5. From the left: i. the profile of the module's wall; ii. extrusion of this profile forming the initial shape **IS**; iii. overlapping of the allowable solid **AS** and initial shape **IS**; iv. module (VZM) resulting from the intersection of solids **AS** and **IS**.

Section 4 documents a number of numerical experiments demonstrating the potential of the automated generation of $Vault - Z$ pavilions of given properties under given constraints. The convergence of the algorithm is discussed, and representative results are visualized through computer renderings. Section 5 provides a discussion identifying the limitations of our method. The article concludes with Section 6, which summarizes the contributions and outlines directions for future and ongoing research.

2. Background

The concept of Extremely Modular System represents a new approach to the design of engineering and architectural structures. The main difference from the traditional modular systems used in engineering, and building construction in particular, is the emphasis of the minimal diversity of types of modules — ideally just one.

2.1. Concept of Vault-Z

$Vault - Z$ is an architectural concept that belongs to the family of Extremely Modular Systems. It is based on one type of module and allows for the creation of free-form vaults of practically any shape. In principle, the $Vault - Z$ module (VZM), is a Boolean intersection of two components: the allowable solid (**AS**) and the initial shape (**IS**). **AS** is the bounding volume enclosing the VZM. It is a three-dimensional intersection of two congruent wedges perpendicular to each other, as explained in Fig. 4.

The angle α of **AS** must be the same in both planes: XY and YZ. While α can take any value from $(0, \pi)$. The most practical values are those derived from divisions of a circle into the desired number of full sectors. This facilitates the creation of closed domes and funnels. In this paper, $\alpha = \frac{\pi}{6}$, therefore a full dome requires exactly 12 modules. Thus, this $Vault - Z$ module is denoted as VZM_{12}.

Fig. 5 illustrates the forming of the module VZM as a result of the Boolean intersection between the allowable solid **AS** and the initial shape **IS**.

Fig. 6 shows examples of simple structures based on the same initial shape (**IS**) and four allowable solids **AS** based on a different angle α.

As Fig. 6 indicates, a wide **AS** (with larger values of α) forms domes with fewer modules. For example, an **AS** based on $\alpha = \frac{\pi}{2}$ forms a dome with only 4 modules ($4\times\frac{\pi}{2} = 2\pi$). On the other hand, an AS based on a smaller angle, e.g., $\alpha = \frac{\pi}{8}$, forms a dome with 16 modules ($16\times\frac{\pi}{8} = 2\pi$).

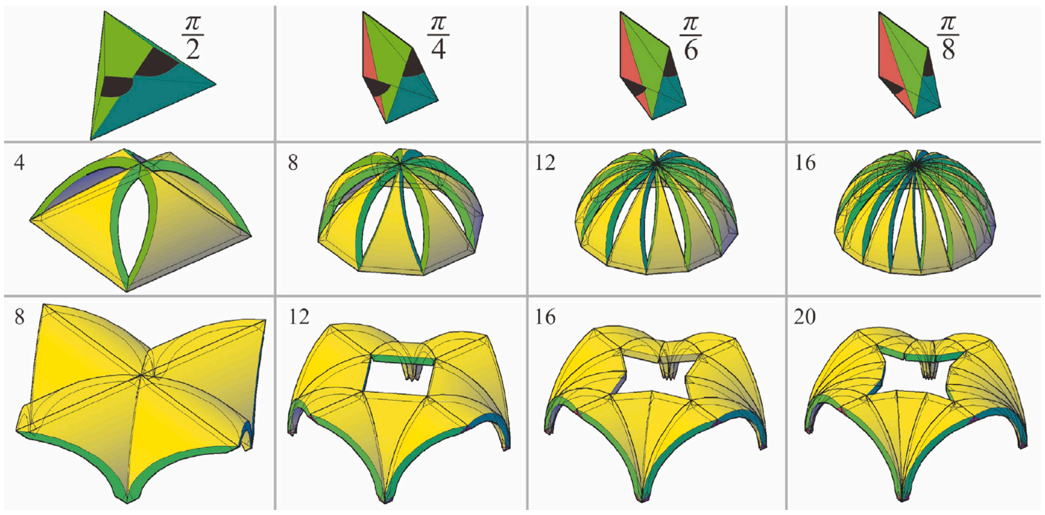


Fig. 6. Four $VZMs$ based on the same **IS** but different **AS** (shown in the top row) based on four different angles α. The middle row shows the corresponding domes with the respective number of modules. The bottom row shows the corresponding groin vaults with the respective number of modules.

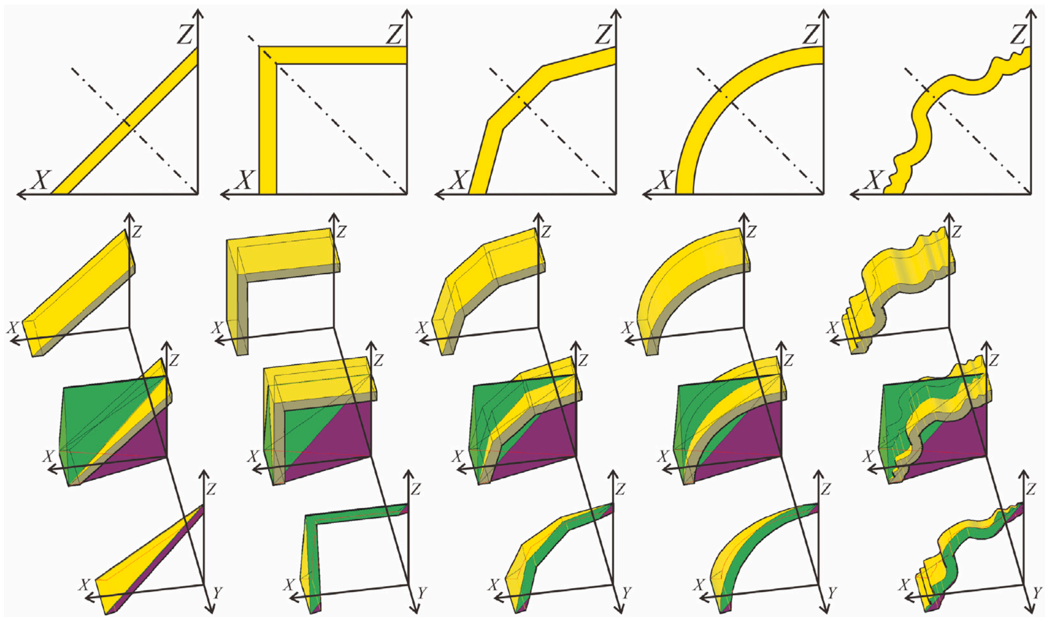


Fig. 7. Five examples of **IS** and corresponding $VZMs$. From left to right: from the simplest profile – a simple straight "shingle" – to a free-form.

The initial shape (**IS**) is equivalent to the wall of VZM, and can take practically any form. It can be created by a simple extrusion, as shown in Fig. 7.

There are two constraints to the shape of **IS**:

1. The profile of the **IS** wall should be symmetrical about the diagonal axis.
2. The **IS** profile must not extend beyond the **AS**, as this would disrupt the continuity of the VZM.

As a consequence of its geometric properties, it is possible to assemble $VZMs$ in four alternative ways, as illustrated in Fig. 8.

As Figs. 6 and 8 indicate, depending on the connection type, there are gaps between main modules: large, medium and none for: △△, △▽, and ▽▽& ◃▹, respectively.

Therefore, the main module is equipped with a pair of side elements. Figs. 9 and 10 present the exploded and assembled drawings of the VZM_{12}, respectively. In these figures, the module is also equipped with a universal opening, which can serve both as a door and as a skylight. This opening is also visible in Fig. 8, which shows the making of the aluminum prototype.

Fig. 11 shows the same four possible connection types presented in Fig. 8 with the side elements.

It is possible to divide a full circle into 12 single VZM_{12}s, but also to group them into: 6 pairs, 4 triplets, 3 quadruples and 2 sextuplets. This allows to efficiently connect 2, 3, 4, 6, and 12 simple $Vault-Z$ domes composed of such VZM_{12}s, as shown in Fig. 12.

Fig. 13 illustrates most of the possible simple multi-domes (except the duodecuple) with 3D visualizations.

Another unique feature of $Vault-Z$ is the ability to create vault intersections (without the top voids as shown in Fig. 12) of different numbers, as illustrated in Fig. 14.

The unprecedented design flexibility of $Vault-Z$ is demonstrated through conceptual emulations of selected contemporary dome systems.

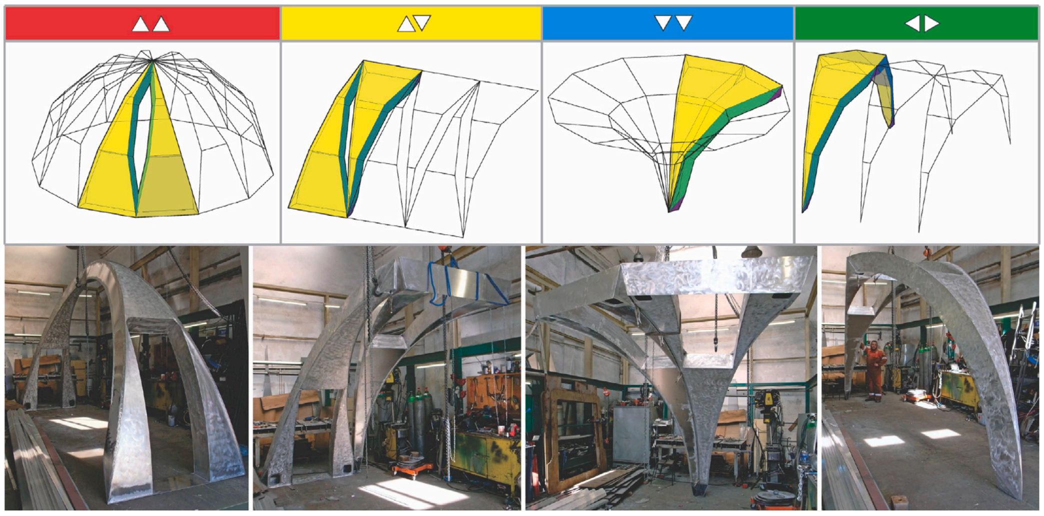


Fig. 8. Top row: four possible connections between a pair of modules. Bottom row: the making of the aluminum prototype modules of $Vault - Z$; from the left: two modules supporting each other (not a valid connection), and three corresponding configurations: △▽, ▽▽ ◁▷. The internal height of a module is 294 cm, while the overall wall thickness is 324 mm.

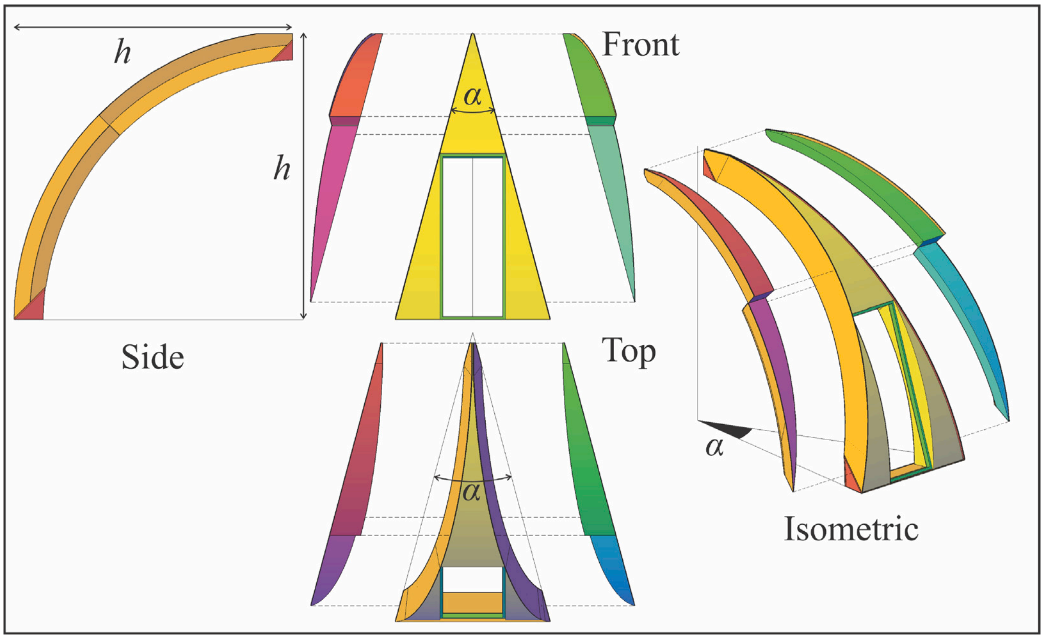


Fig. 9. Exploded view of the $Vault - Z$ module (VZM_{12}, since $\alpha = \frac{\pi}{6}$) and side elements (red and green indicate: left and right sides, respectively).

2.1.1. *Contemporary dome architecture: Example 1*

Several home floor plans are available at the Monolithic Dome Institute [10]. For example, “Partial Torus”, shown in Fig. 15 along with its $Vault - Z$ triangulation and visualization.

As Fig. 15 illustrates, $Vault - Z$ allows to emulate this toric floor-plan quite closely. Moreover, certain modular apertures can be opened or closed to fit the functional requirements of the interior.

2.1.2. *Contemporary dome architecture: Example 2*

Another example is “Triton”, which is an elongated composition of three domes. Fig. 16 shows this floor-plan along with its $Vault - Z$ triangulation and visualization. As Fig. 16 illustrates, $Vault - Z$ also facilitates the emulation of floor-plan based on three domes. As in the previous example, the modular apertures, to a certain degree, reflect the functionality of the interior.

Fig. 17 shows an emulation of a *Bolwoningen* housing unit (described in Section 1) by the $Vault - Z$ system. A total of 23 out of 24 regular modules are used. Only one module is trimmed to make an aperture for the main entrance.

2.1.3. *Prototype*

Presently, an experimental full-scale $Vault - Z$ prototype is under construction. Twelve aluminum modules for a dome pavilion with a usable surface area of approximately 35 m^2, and a height of about 3 m have been delivered to the site (see Fig. 18).

This pavilion will serve multiple purposes: a proof-of-concept exhibition, an experimental site for determining the thermal and ventilation characteristics of the units, and a platform for experimenting with the functional properties of the system (see Fig. 19).

2.2. *Optimization of structures*

This work also fits into a broader, active body of research that explores how soft computing methods, particularly genetic algorithms

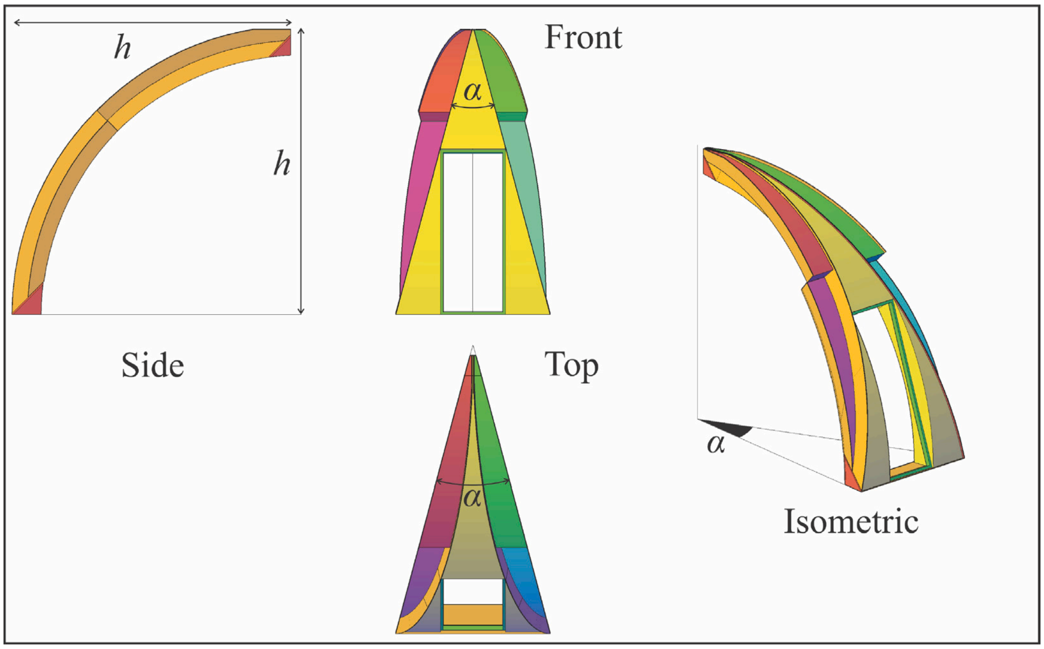


Fig. 10. Views of the VZM_{12}. Color convention as in Fig. 9.

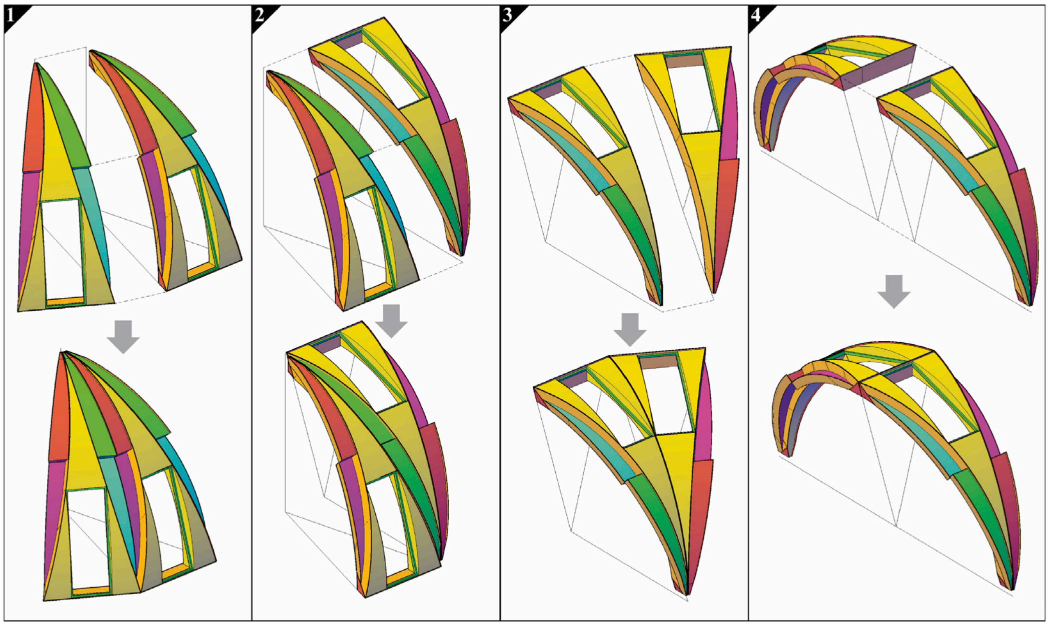


Fig. 11. Four connection types accounting for the side elements. (1) △△: the sides of two adjacent modules come together. (2) △▽: the side elements slide over each other. (3) ▽▽: in this case, the side elements must be removed. (4) ◁▷.

and other nature-inspired metaheuristics, can be applied to early-stage architectural design. The idea and the framework presented here should be viewed as one contribution to the rapidly expanding line of research on AI-assisted conceptual design identified by [11]. Their review shows that most early-stage work still relies on evolutionary search, from classic floor-plan systems such as EvoArch [12] to optimization studies in sustainable building form and envelope design [13]. Our method advances this field by (i) adopting a graph-based genotype that captures modular topology, (ii) providing domain-specific genetic operators, and (iii) embedding an interactive weight-tuning workflow so designers can explore the ambiguous goal space highlighted by the review. In this way, the study complements existing EA-driven layout engines and performance-oriented optimizers while addressing the open call for transparent, designer-in-the-loop tools for conceptual exploration.

The following subsections describe the mathematical methodology applied to the generation of optimal and near-optimal $Vault-Z$ structures.

2.2.1. *Problem statement and complexity*

Given a number N of modules VZM, a modular structure can be initiated by placing the first VZM and attaching a second one to it by selecting one of the valid connection types (as described in Section 2.1). Subsequent modules can be attached in various ways, provided that structural constraints are not violated — for example,

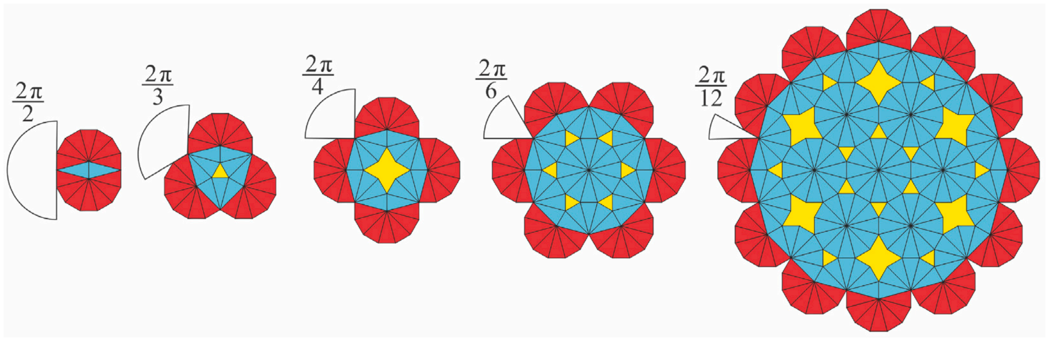


Fig. 12. Connecting 2 (double), 3 (triple), 4 (quadruple), 6 (sextuple), and 12 (duodecuple) partial $Vault - Z$ domes. Red indicates modules in the $\triangle$ position – i.e., partial domes. Blue indicates modules in the $\bigtriangledown$ position – i.e., (partial) funnels. Yellow indicates voids.

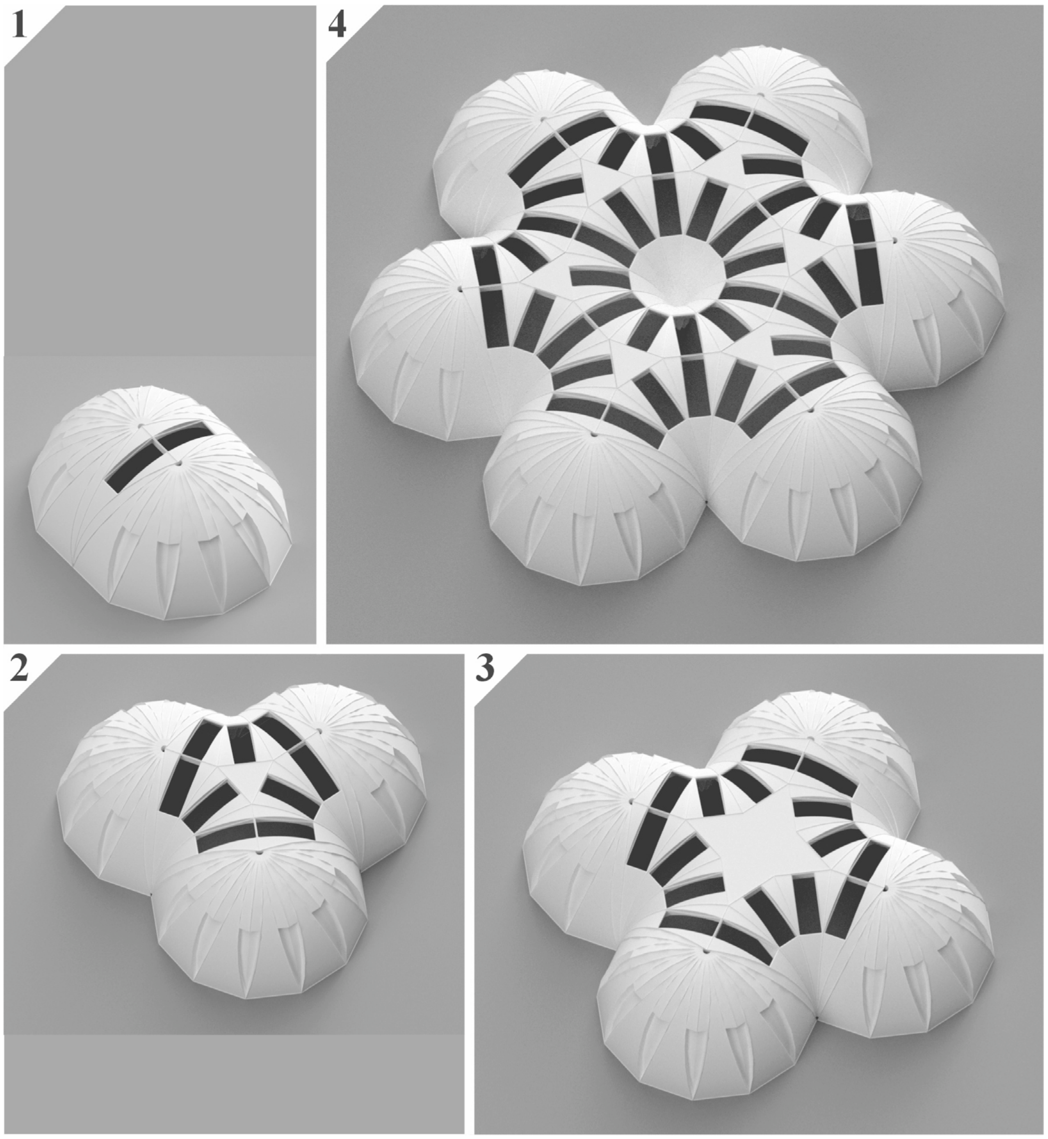


Fig. 13. $Vault - Z$ multi-domes: 1. Double dome. 2. Triple dome. 3. Quadruple dome. 4. Sextuple dome. The apertures in all $\triangle$ modules are closed, all apertures in $\bigtriangledown$ are open. The voids (shown in yellow in Fig. 12) are covered with additional panels.

newly attached modules must not physically collide with the existing structure. Once all N modules are connected, a configuration C is obtained, corresponding to a valid $Vault - Z$ layout.

Naturally, humans can intuitively find certain classes of such configurations; for example, constructing a dome or a barrel vault is relatively straightforward. For moderately larger structures, the assembly becomes a combinatorial puzzle that can still be solved manually. However, as the number of modules increases, finding optimal (in some sense) configurations becomes increasingly challenging due to the astronomical number of possible combinations.

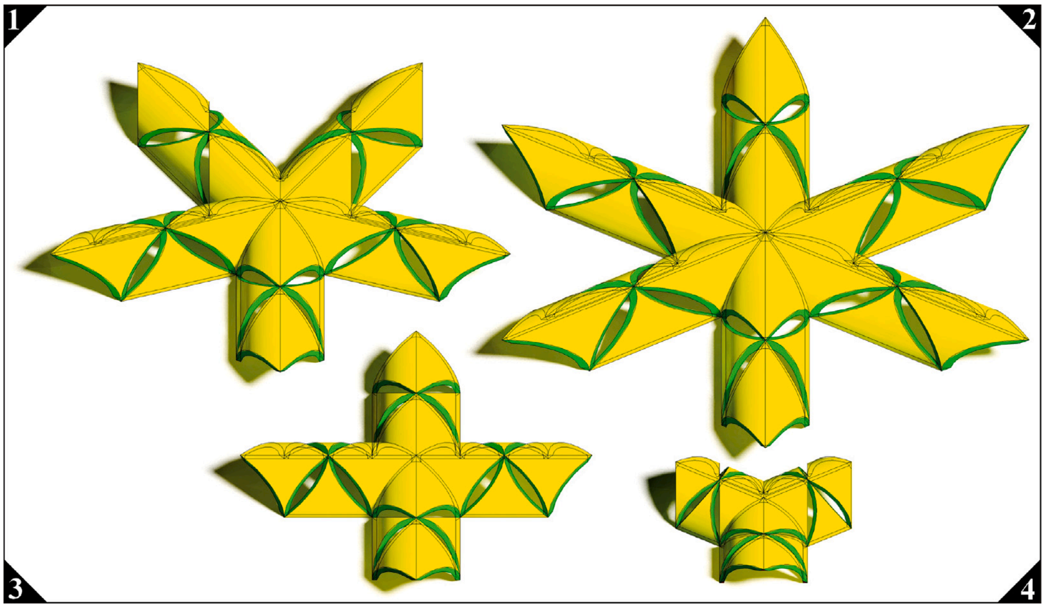
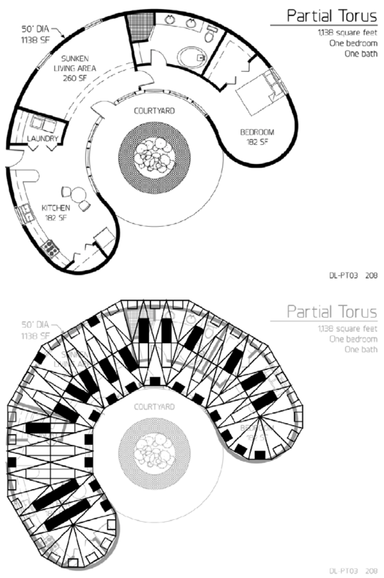


Fig. 14. $Vault - Z$ multi-vaults: 1. Pentapartite (angle $\alpha = \frac{3\pi}{5}$). 2. Hexapartite $\alpha = \frac{2\pi}{3}$. 3. A regular groin (quadripartite) vault $\alpha = \frac{\pi}{2}$. 4. A tripartite vault $\alpha = \frac{\pi}{3}$. For clarity, the side elements are not shown.



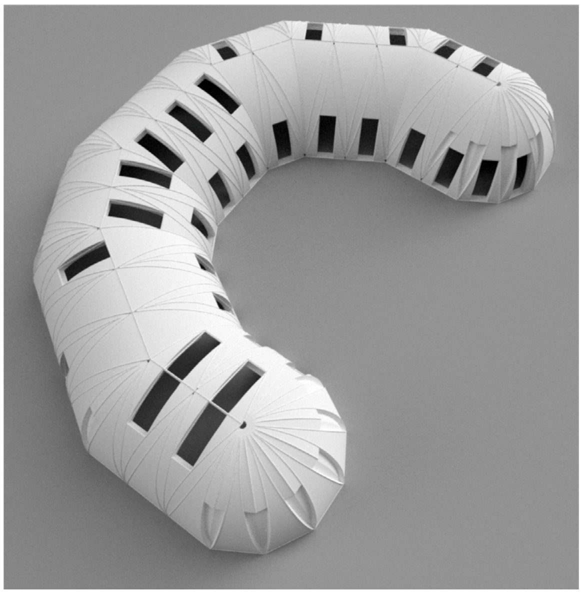


Fig. 15. On the left: Top: Original plan of *Partial Torus* ©Monolitic Dome Institute. Bottom: "Triangulation" with $Vault - Z$ modules. On the right: 3D visualization.

To discuss any general optimization process, it is necessary to define a formal method of evaluating each configuration C by introducing a function $f(C) \in \mathbb{R}$ [14]. At this point, we do not make any assumptions about the form of $f(C)$, and require only that it be numerically evaluated for any given configuration. This can be used for ranking any set of considered $Vault - Z$.

Formally, the goal of the optimization is to find a configuration $C^* \in \mathcal{S}$ that maximizes (or minimizes) the objective function:

$$C^* = \arg\max_{C \in \mathcal{S}} f(C). \tag{1}$$

Each configuration C belongs to the space $\mathcal{S}$ of all possible configurations for a given number of modules N. As N increases, the configuration space expands exponentially, with each additional module introducing multiple non-trivial placement constraints. For small values of N, it is possible to determine the number of unique feasible configurations through direct brute-force simulations [15]. We have enumerated all unique valid configurations by considering every allowable connection, up to $N = 8$ modules. For example, for $N = 7$, there are exactly 3 437 unique ways of connecting the modules.

The number of possible configurations, as determined by the calculation above, clearly exhibits exponential growth (see Fig. 20). This allows for the formulation of an empirical scaling law estimating the number of valid configurations, approximately:

$$N_{\text{poss.conf}} \approx 0.22 \cdot 10^{0.6N} \tag{2}$$

The data points obtained through exhaustive enumeration were fitted using nonlinear regression (on a semi-logarithmic scale), demonstrating the explosive combinatorial growth of the configuration space [16]. This fitting enables extrapolation to estimate the size of the search space $\mathcal{S}$ for larger values of N [17].

For example, for $N = 12$ modules, there will be approximately 4 million configurations, while for $N = 48$ modules, this number reaches

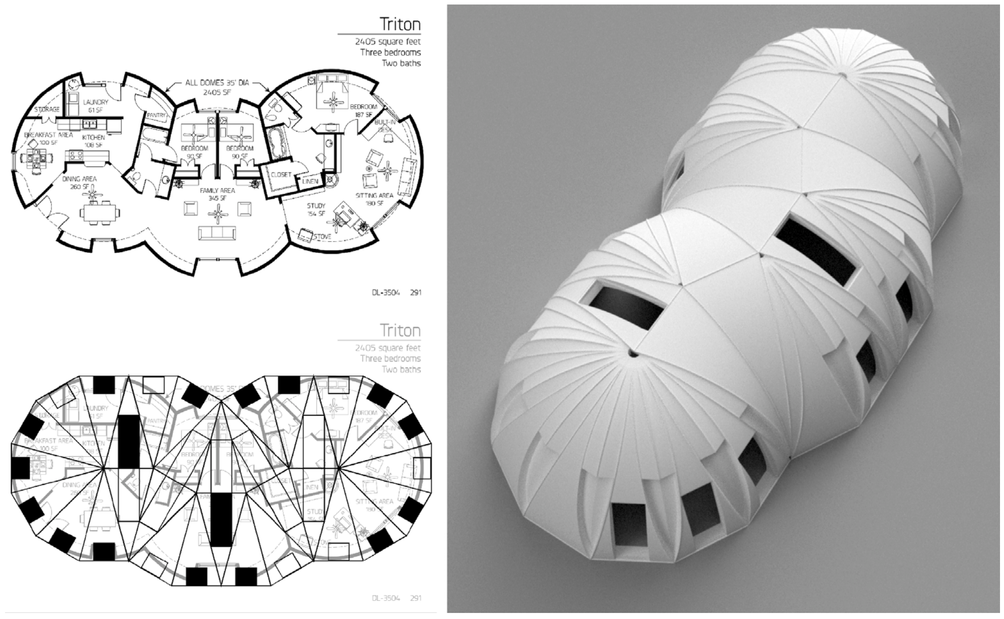


Fig. 16. On the left: Top: Original plan of *Triton* ©Monolitic Dome Institute. Bottom: “Triangulation” with $Vault - Z$ modules. On the right: 3D visualization.

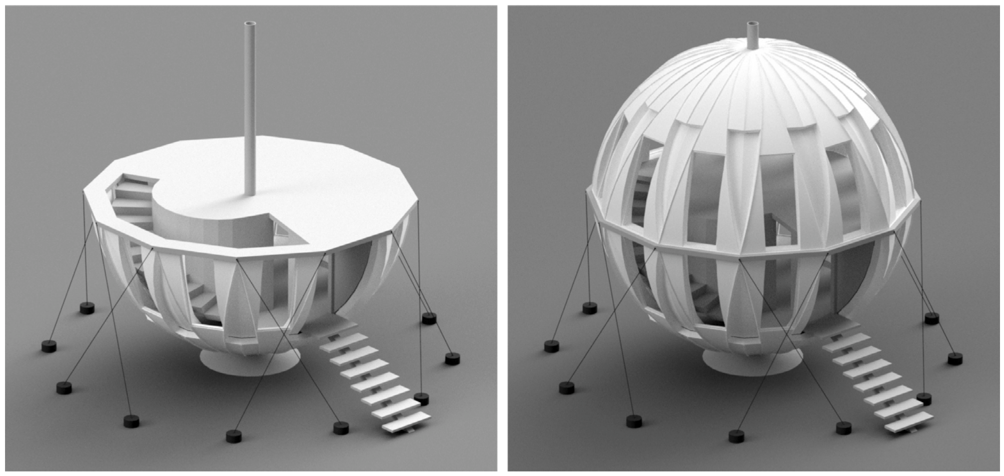


Fig. 17. A two-story “ball house” made with 24 $Vault - Z$ modules. On the left: the ground floor showing the internal spiral stairs. On the right: the complete spherical unit. The door/window elements can be glazed or filled, according to the design requirements.

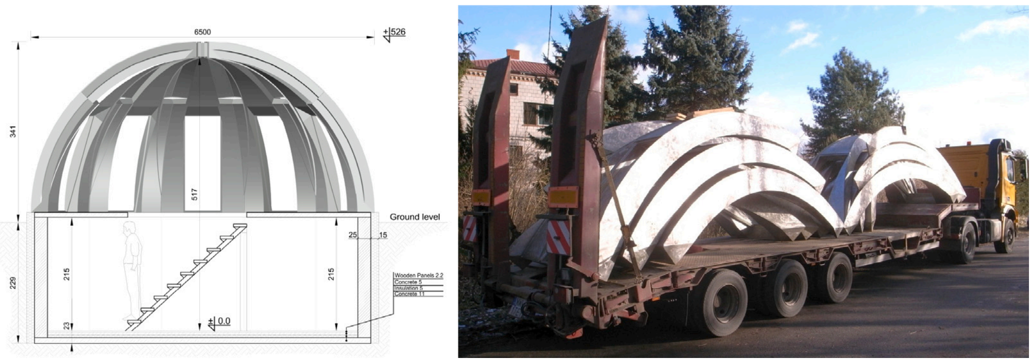


Fig. 18. Prototype of the $Vault - Z$ dome. On the left: the cross-section. On the right: the delivery of 12 prefabricated main modules (without the side elements and glazing) each weigh approximately 75 kg.

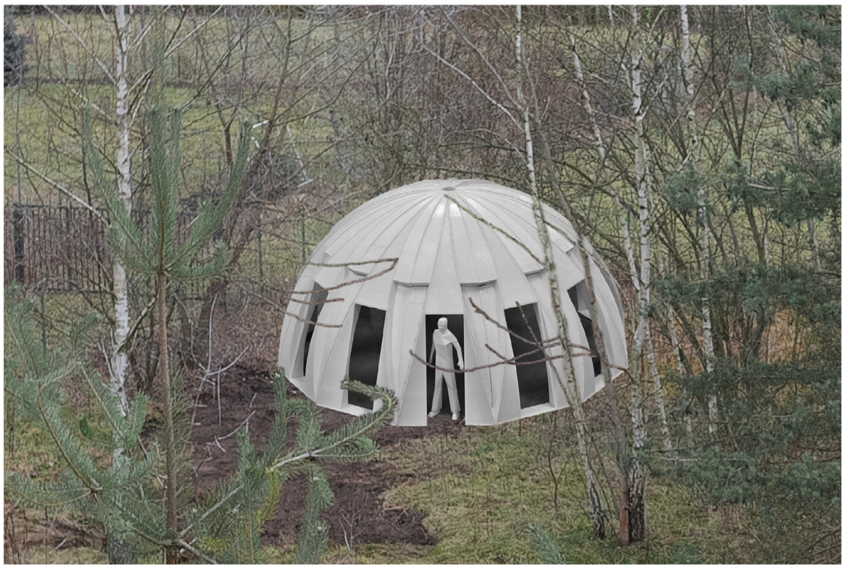


Fig. 19. Visualization of the prototype $Vault - Z$ dome in its site setting.

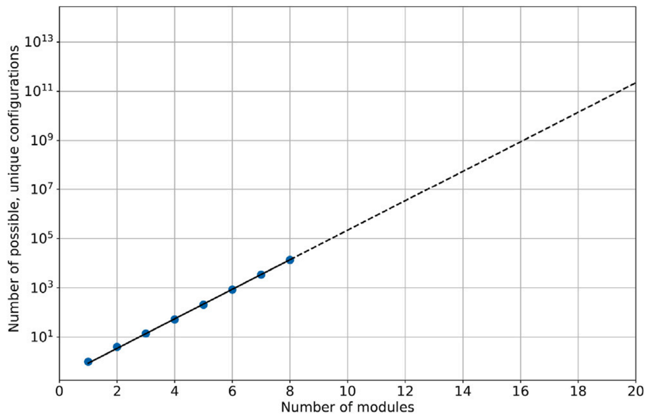


Fig. 20. Number of possible ways to connect N modules into unique structures as a function of N. The dots represent exact numbers found through computational methods, and the line represents a fitted scaling law $0.22 \cdot 10^{0.6N}$.

about $2.8 \cdot 10^{28}$ (see Fig. 20). Due to this exponential growth, any brute-force optimization approach – where each possible configuration is explicitly evaluated – becomes clearly infeasible, as it would require computing $f(C)$ for all $C \in \mathcal{S}$.

2.2.2. Application of an evolutionary algorithm to the optimization problem

The most natural representation of a configuration C – composed of connected modules – is a graph G. In this graph, each node represents a module, and each edge denotes a specific way of connecting two adjacent modules (details are provided in Section 3). As a result, the optimization process can be formulated as a combinatorial problem, where the objective is to find a graph G^* that maximizes the fitness function.

Obviously, many combinatorial problems can be expressed using graph structures and solved with a variety of well-established algorithms. In numerous cases, a given problem can be shown to be equivalent to a classical or generalized variant of a well-known problem, such as the Graph Coloring Problem (GCP) [18], the Vehicle Routing Problem [19,20], or the Traveling Salesman Problem (TSP) [21,22]. These problems are typically NP-hard [23], and a vast number of exact and approximate algorithms have been developed to address them.

However, the optimal construction of a $Vault - Z$ configuration cannot be directly formulated as any of the classic graph optimization problems for two main reasons:

- The underlying directed graph cannot be easily reduced to a standard representation (such as a path or permutation, as in TSP and related problems [24]).
- The definition of the fitness function must be as general and *flexible* as possible. It must accommodate diverse architectural objectives, including rewards or penalties based on environmental interactions, such as irregular terrain height maps or physical obstacles. These factors influence the graph but are not inherently part of its structure.

This requirement for general applicability suggests the use of a meta-heuristic approach [25] – specifically, Evolutionary Algorithms (EAs) [26]. EAs mimic natural processes such as selection, mutation, and crossover. They have been successfully applied to the most challenging computational puzzles — namely, problems that are solvable in nondeterministic polynomial time (NP-hard). EAs are widely applied in computer science, automation, operations research, and other fields [27].

Our motivation for using EAs stems from evidence that EA-based methods perform well on many classical combinatorial problems, especially when exact solutions are impractical for larger instances (the Traveling Salesman Problem and its variants, Vehicle Routing Problem, and many more [28]). This approach recently gained significant momentum for solving conceptual design problems in architecture, as was demonstrated in the review article [11]. The authors show how EAs – and other methods from the field of artificial intelligence; evolutionary computing (EC) in particular – profoundly change the way in which the complicated processes of architectural design are approached. A successful application of EAs for optimization of building shading based on cellular automata, which served both: daylight control and aesthetic functions was presented in [29]. EAs were also an effective optimization tool in the case of the earlier Extremely Modular System –*Truss-Z* – a self-supporting free-form footbridge, both for single-branch [30], and multi-branch structures [31].

A major advantage of using EAs for this problem is human control over the fitness parameters and the ability to generate a diverse set of near-optimal solutions. This provides the designer with multiple *good* structures (i.e., those with high fitness values) to choose from. As a result, it becomes possible to consider configurations not fully captured by the formal fitness function, such as aesthetic preferences [32,33], which are inherently subjective.

This approach aligns well with the goals of interactive evolutionary computation (IEC). Typically, EAs require formal identification of the suitability criteria and then search for some optimum solution. Interactive methods also use subjective input from users which enters the search process – here the user input is realized by tuning parameters of the fitness function and subjective selection of individuals from results of multiple trials. IEC has proven highly effective in solving a wide range of real-world problems that are difficult to quantify mathematically or poorly suited to traditional computational models. Its success comes from the ability to combine EC with expert knowledge and user preferences. Applications include product design, industrial planning, art design, image generation, sound composition, and many more [34,35].

2.2.3. Representation of individuals

In any evolutionary algorithm, a candidate solution must be encoded in a format that can be stored in memory and manipulated by genetic operators acting on a population. Designing an appropriate representation – along with corresponding operators – is a crucial step, as it can profoundly influence the effectiveness of the optimization process [36,37]. A good representation should not only cover the entire search space but also guide the search efficiently towards high-quality solutions, thereby preventing the algorithm from degenerating into a random search.

The most common types of representations include binary strings, integer or real-valued vectors, permutations, and trees. In the context of the modular structure optimization problem considered here, two main options exist: i. transform the graph into an alternative encoding suitable for a standard EA framework; or ii. define genetic operators that act directly on the graph representation G of configurations $C \in S$.

Direct manipulation of graph structures in evolutionary algorithms has been explored in prior work, though it remains significantly less common than more traditional representations. For instance, the graph-based evolutionary algorithms introduced by [38] use a geographical-inspired approach to manage population diversity and enhance evolutionary algorithms. In a recent work [39] regarding a very different topic – molecular design – the authors demonstrate how EAs can be used in the process of designing molecules. In this case, molecules are represented as graphs, and the entire process is supported with machine learning surrogate models which control population diversity and enable faster fitness function evaluation. A graph-based genetic algorithm designed for such a purpose has been implemented in a popular open-source cheminformatics package, RDKit [40]. Additionally, recent research in genetic programming shows that graph-based representations can naturally facilitate code reuse, leading to more compact and expressive solutions [41,42]. Another interesting direction worth mentioning is the integration of Graph Neural Networks (GNNs), where a genetic algorithm evolves a single population-level graph [43]. In this approach, each candidate solution in the population is represented as a node in a graph, and the adjacency matrix encodes similarity between individuals. This representation enables the algorithm to leverage global correlations among individuals, effectively guiding the search process by propagating structural information across the population.

Graphs offer several advantages as a representation for individuals in EAs:

- They offer a natural and intuitive encoding for structural and relational problems, such as those found in architecture, circuit design, or modular assemblies.
- They can represent arbitrary relationships – not just linear or hierarchical structures – as well as structures involving reuse and cycles, which are difficult to express in vectors, trees, or permutations.

However, graphs also introduce challenges:

- Encoding, mutation, and crossover require problem-specific logic to ensure structural validity.
- Maintaining validity after genetic operations can be computationally expensive.
- Small modifications (e.g., adding or removing an edge) may lead to large, nonlinear changes in the phenotype—although this issue can affect other representations as well.

In the case of *Vault-Z* structures, we can reasonably expect that any useful configuration forms an enclosed system. This implies that the corresponding graph representation will contain cycles. Such cycles tend to form substructures that can be reused during the evolutionary process across generations. This is a key advantage over more traditional encodings, where such architectural invariants would be harder to preserve under recombination or mutation. With alternative representations, the search process would likely become more random and less structurally informed, which provides strong motivation for employing a dedicated graph-based evolutionary optimization approach.

Based on the above considerations, our optimization framework is built upon the following three essential components:

- A graph representation of $Vault-Z$ that ensures structural validity;
- A meta-heuristic evolutionary algorithm with genetic operators applied directly to the graph (selection, mutation, and crossover);
- A *flexible* and efficient fitness function capable of guiding the search towards desirable architectural solutions.

3. Graph-based optimization framework

Given the geometry of a single VZM module, the main objective is to find a way of assembling them into a structure that meets the optimization criteria. To achieve this, a formal description of the connections of modules into a structure is required. Here, a directed graph is proposed for this purpose: nodes represent the VZMs, and edges define the connections between them. Since there are various ways of connecting VZMs, each link is associated with its specific geometric transformation. The graph of connections (transformations) is general, and can represent any structure composed of VZMs.

$Vault-Z$ can form fully three-dimensional shapes, such as the faceted sphere shown in Fig. 17. However, for simplicity, the optimization technique presented here only considers planar arrangements, where all VZMs are placed on a flat surface and maintain at least one point of contact with it. This constraint allows each feasible $Vault-Z$ configuration to be uniquely represented by its two-dimensional projection. Therefore, there exists a one-to-one correspondence (bijection) between the spatial $Vault-Z$ structure and its planar representation, as illustrated in Fig. 21. For further examples of projected layouts and their spatial counterparts, see also Figs. 12 and 13.

As Fig. 21 indicates, each VZM_{12} can be placed in the upright (△) position, i.e., the base of the module is placed on the ground, and reversed (▽) position, where the module is placed on its tip. From a structural point of view, it is disadvantageous when the base of a module does not connect to the ground or another module. This is represented in the 2D projection by a situation where the shorter side of a blue triangle connects to the white background, e.g., module 10 in Fig. 21. Such configurations are called "open sides" and penalized in the fitness function (Eq. (3)) as N_{open}.

In order to accelerate the calculations, the algorithm operates on two-dimensional layouts. One main advantage is that collision detection is substantially simpler for planar triangles than for complex 3D curved solids.

The construction of graph G for each individual begins with a single node n_0 (the first module of a $Vault-Z$) placed in the upright

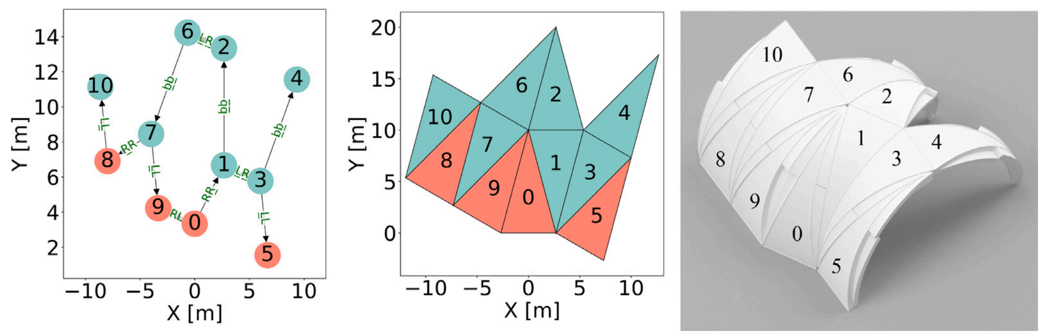


Fig. 21. On the left: An example of a directed graph G. In the middle: The corresponding projection of $Vault - Z$ represented as triangles. On the right: The visualization of the actual $Vault - Z$. The upright (△) and reversed (▽) modules are shown in red and blue, respectively.

position (△). Subsequent modules can be connected to its right (R) or left (L) side in the upright (△) or reversed (▽) orientation. For the VZM orientation nomenclature, refer to Figs. 9 and 10. The naming convention used here denotes reversed modules with an underlined letter. If a VZM is in a reversed orientation (▽), it is also possible to attach another VZM via their bases (b) (see Figs. 8 and 11.4). All types of connections (graph edges) are listed in Table 1, and an example is shown in Fig. 21. A graph G constructed in this way is directed, as each edge $e_{A,B}$ represents a one-way transition from module A to module B. However, an equivalent connection type always exists from B to A, therefore $e_{A,B}$ can be replaced by $e_{B,A}$, resulting in the same structure. This property has no effect on the algorithm.

Each valid graph G represents a $Vault - Z$ layout, where triangles are planar projections of VZM; that is:

- The triangles do not intersect, although they may share edges.
- If two VZMs are physically connected by sharing an edge, this must be reflected as a respective edge in the graph G;
- Only one triangle may be attached to a side of another triangle.

The fitness function evaluates the planar layouts, which are projections of $Vault - Z$s placed within a given environment on a 2D surface. Such environments impose constraints — for example, obstacles, lot size and shape. Therefore, it is necessary to project a graph G, which, in principle, is a set of topological relationships among nodes, on the given environment. This is done as follows:

- A node of the graph G is selected as the "anchor", and denoted as n_a.
- Next, the planar layout of $Vault - Z$ is generated from graph G.
- Finally, this layout is placed at $(0, 0)$ of the planar projection of the given environment by the anchor node n_a and oriented so that the base of the corresponding triangle is parallel to the X axis

3.1. Fitness function

The fitness function assesses the quality of potential solutions and directs the algorithm towards optimal results. Such a function must be efficiently computable and well-aligned with the optimization objectives. Fitness function formulation is particularly challenging in architecture-related problems, where the criteria are often unclear and conflicting.

A fundamental parameter that limits the size of $Vault - Z$ is the maximal number of available modules ($N_{\max}$). This does not imply that all of them must be used in the final solution. For example, structures composed of $N = 12$ VZM_{12} will generally have better fitness than those of $N = 13$.

Each VZM_{12} encloses a certain amount of space, which corresponds to the projected area on the ground surface. One of the objectives is to maximize the total area A_{tot} covered by the structure. At the same time, it is undesirable to leave any "open sides", i.e., modules with unused connecting sides. Their number is denoted N_{open}. Since open $\underline{\text{bb}}$ connections are generally "more tolerable" from an architectural perspective, they are counted as $1/3$ (weaker penalty). Finally, in order to make the structure enclosed, the number of nodes in any simple cycle (N_{cycl}) in the graph should be maximal. Rewarding cycles is beneficial in many graph optimization methods (e.g., [24]). Here, cycles correspond to closed structures, which intuitively lead to better solutions with fewer open sides.

The fitness function incorporating only the criteria listed above would not distinguish between tunnel-like solutions (see Fig. 23) and other configurations that may be more suitable for real-life applications or aesthetically preferable from a designer's perspective. Therefore, additional criteria were implemented:

- The ratio between the area A_{circ} of a circle enclosing all modules to the area A_{tot};
- The total area fully enclosed by the structure A_{enc};
- Avoidance of collisions with existing environmental obstacles.

The fitness function f used in this study is defined as follows:

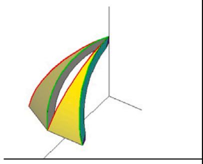
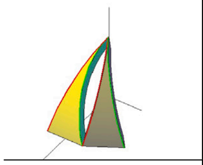
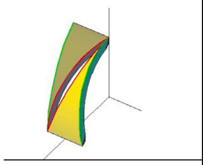
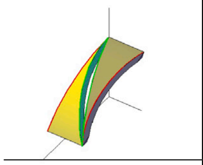
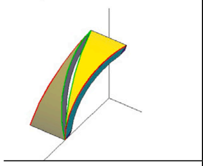
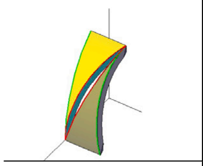
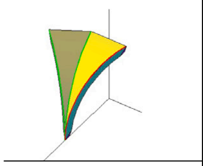
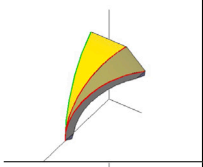
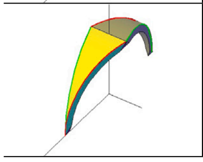
$$f = \left(\frac{A_{\text{tot}} + \zeta A_{\text{enc}}}{A_1}\right)^{\alpha} \left(\frac{N - N_{\text{open}}}{N}\right)^{\beta} \left(\frac{N_{\text{cycl}}}{N}\right)^{\gamma} \left(\frac{A_{\text{tot}}}{A_{\text{circ}}}\right)^{\delta} \tag{3}$$

where N is the number of nodes; A_1 is the surface area of a single module; parameters α, β, γ, and δ adjust the importance of: maximization of the surface area, minimization of the number of open sides, maximization of the number of nodes belonging to a cycle, and resemblance to a circular shape (roundness), respectively; parameter ζ adjusts the importance of maximization of the enclosed area by the structure.

Encoding architectural goals into a fitness function is inherently problematic because such goals are often qualitative, multi-layered, and context-dependent, making them difficult to reduce to single, scalar values. The choice of weights or metrics reflects subjective design priorities that may shift over time or differ between architects, challenging reproducibility and universality. In the discussed case, such subjective choices are basically reduced to setting values of α, β, γ, and δ. The choice for these parameters depends on particular goals and environmental constraints. Each raw objective in the fitness definition, Eq. (3), is first normalized so that its magnitude is close to unity for a reasonable design. The exponents therefore only express relative importance; they do not alter units or introduce hidden bias. Setting a weight to zero removes that objective entirely, which is a deliberate feature that allows designers to explore single-objective runs. Both binary and continuous weights are permitted, and, for example, the choice $\alpha = \beta = \gamma = \delta = 1$ is purely arbitrary and can be made to demonstrate the outcome of optimization where the penalty terms are treated equally.

Regarding the question – "how can a user decide in advance which values are the most suitable for a specific structural application?" – there is no simple answer (this is a common problem, see e.g. [11]). Our

Table 1
Edge types in the graph G representing valid connections between two VZM_{12}. Module A and B are shown in yellow and gray, respectively. Right and left edges are indicated by green and red lines, respectively.

Visualization	A	Edge	B	Description
	$\triangle$	LR	$\triangle$	Module A, in the upright orientation has attached to its left side (L) module B in the upright orientation and by its right side (R).
	$\triangle$	RL	$\triangle$	Module A, in the upright orientation has attached to its right side (R) module B in the upright orientation and by its left side (L).
	$\triangle$	L$\underline{\text{L}}$	$\triangledown$	Module A, in the upright orientation has attached to its left side (L) module B in the $\underline{\text{reversed}}$ orientation and by its left side ($\underline{\text{L}}$).
	$\triangle$	R$\underline{\text{R}}$	$\triangledown$	Module A, in the upright orientation has attached to its right side (R) module B in the $\underline{\text{reversed}}$ orientation and by its right side ($\underline{\text{R}}$).
	$\triangledown$	$\underline{\text{R}}$R	$\triangle$	Module A, in the $\underline{\text{reversed}}$ orientation has attached to its $\underline{\text{right}}$ side ($\underline{\text{R}}$) module B in the upright orientation and by its right side (R).
	$\triangledown$	$\underline{\text{L}}$L	$\triangle$	Module A, in the $\underline{\text{reversed}}$ orientation has attached to its $\underline{\text{left}}$ side ($\underline{\text{L}}$) module B in the upright orientation and by its left side (L).
	$\triangledown$	$\underline{\text{RL}}$	$\triangledown$	Module A, in the $\underline{\text{reversed}}$ orientation has attached to its $\underline{\text{right}}$ side ($\underline{\text{R}}$) module B in the $\underline{\text{reversed}}$ orientation and by its left side ($\underline{\text{L}}$).
	$\triangledown$	$\underline{\text{LR}}$	$\triangledown$	Module A, in the $\underline{\text{reversed}}$ orientation has attached to its $\underline{\text{left}}$ side ($\underline{\text{L}}$) module B in the $\underline{\text{reversed}}$ orientation and by its right side ($\underline{\text{R}}$).
	$\triangledown$	$\underline{\text{bb}}$	$\triangledown$	Module A and B, in the $\underline{\text{reversed}}$ orientation, are attached to each other by their bases ($\underline{\text{b}}$).

procedure is to set all the exponents to 1 and then decrease or increase the exponents which correspond to a particular property (roundness, internal enclosed area, etc.). Alternatively – if resources allow – one can run a Latin-hypercube sweep covering a range of parameters and compare the best individuals.

To solve the optimization problem, i.e., to maximize f, we implemented a dedicated evolutionary algorithm tailored to the graph-based representation of $Vault-Z$ structures. While the overall structure of the algorithm follows standard evolutionary computation principles – initialization, selection, crossover, mutation – the genetic operators themselves are custom-designed to operate directly on graph-based individuals. The framework is implemented in pure Python with DEAP [44], with collision checking using a dedicated library (PyMesh). In principle it can be significantly optimized. A single run on an i7 at 3.6 GHz with $N_{\max} = 50$, $N_{\text{popul}} = 100$ and$T = 100$ generations takes about 2 min. Trials can be run in parallel on a multi-core machine or cluster.[1]

3.2. Evolutionary algorithm

To solve the optimization problem defined above, we employed an evolutionary algorithm with dedicated operators acting on a graph. The population consists of N_{popul} individuals (i.e., graphs G). Each generation comprises the following steps:

- Mutate each individual with probability P_{mutate}
- Crossover two selected individuals with probability P_{cx}
- Tournament selection with tour size of 2.

Initially, all individuals (graphs G) consist of single node that gradually grows through mutation. The population is evolved for $T_{\max}$ generations. Selection is performed using *tournament selection*, a standard method in evolutionary algorithms [37]. In this approach, a small number of individuals (e.g., two) are randomly selected from the population, and the individual with the higher fitness value is chosen to reproduce.

3.2.1. Mutation

Mutation is applied by performing one of the following actions, each with equal probability $1/3$ (the degree of modification is specified by a random number N_{mod}, $1 \leq N_{\text{mod}} \leq 0.1N$):

- Grow the graph by attaching N_{mod} random nodes. The grown graph must be valid.
- Remove N_{mod} randomly selected nodes; after removal, rebuild the graph to ensure validity.
- Remove N_{mod} randomly selected “leaf” nodes (i.e., nodes with unused connections).

3.2.2. Crossover

Crossover between two individuals G_1 and G_2 is performed by applying one of two operators (each with probability $1/2$):

(A) Let $e1_{\text{out}}$ and $e2_{\text{in}}$ represent sets of possible out-edges of G_1, and possible in-edges of G_2, respectively. A possible out-edge is randomly chosen for G_1 and for this a matching in-edge from G_2 is drawn as well. Then the merged graph is reconstructed starting from a random node, and the graph is returned as new G_1. The same procedure applies for the new G_2.

(B) As above, but each time the graph is reconstructed from its anchor node.

If graphs G_1 or G_2 lack available out- or in-edges, the crossover operator does nothing; therefore, a prior mutation is required to break a fully closed graph for the subsequent possibility of a crossover.

The reconstruction of the entire graph – after a mutation or crossover – is crucial, since it ensures that the graph is valid. It starts at a given node and involves breadth-first traversing, including predecessor nodes. Nodes are only attached if the graph remains valid, i.e., without self-intersections, with solely valid connections and the total number of nodes $N \leq N_{\max}$. Thus, nodes are renumbered after the reconstruction and the node from which it starts becomes the graph's starting node (in our formulation, the anchor node is the same as the starting node; generally this does not have to be the case). In the (A) case, the starting node is chosen randomly after the graphs are merged. Consequently, it is possible that one or both anchor nodes from the parent individuals will not be included in the offspring at all. Additionally, since the anchor node in the offspring is chosen at random, its location in the environment will be changed. On the other hand, in the (B) version of the operator, the graph is always reconstructed from the anchor node of the first individual. This assures that this node remains the same after the crossover process. While this is irrelevant when looking for solutions not embedded in any environment, it does matter when anchoring in a coordinate system. This helps to keep individuals in the population that have valid anchorage; meaning that the graph does not collide with obstacles.

An example of crossover operation is shown in Fig. 22.

As illustrated in Fig. 22, the parents G_1, G_2 are shown at the top. The offspring $\text{cx}(G_1, G_2)$ results from attaching node 0 from G_1 to the node 9 in G_2 by $\text{R}\underline{\text{R}}$ connection. Then the graph is reconstructed from node 7 in G_1 (so the new anchor node 0 in $\text{cx}(G_1, G_2)$ corresponds to the node 7 in G_1). $\text{cx}(G_2, G_1)$ is a result of attaching node 6 from G_1 to the node 9 in G_2. The new graph is reconstructed starting from its anchor node 0, i.e., variant (B) of the operator. In both cases, the resulting graphs are larger (18 and 17 nodes) than G_1 and G_2 (9 and 10 nodes, respectively). Obviously, this does not always have to be the case. The size of the new graphs is constrained by the value of $N_{\max}$ and possibilities of valid connections.

An example result from a simple optimization, which does not impose restrictions on the $Vault-Z$ shape and assumes no environmental obstacles, is shown in Fig. 23. The following parameters were used: $N_{\max} = 48$, $\alpha = \beta = 1$, and $\gamma = \zeta = 0$. In this example, all nodes belong to a cycle; there are no open sides, and all available modules are utilized. It is relatively straightforward to rearrange some of the modules to form a linear tunnel. This configuration could also be easily extended by an additional four modules. In contrast, achieving compact, circular-like solutions for a given $N_{\max}$ is considerably more challenging. This topic is discussed in the following Section 4.

4. Results

The effectiveness of the proposed approach is demonstrated with several examples that capture a variety of scenarios. For simplicity, unless explicitly stated otherwise, the parameter values are set as follows: $\alpha = \beta = \gamma = 1$, with either non-zero δ or non-zero $\zeta = 1$, depending on the kind of required geometric confinement. In all experiments, the population size is $N_{\text{popul}} = 200$, with a mutation probability of $P_{\text{mut}} = 0.3$, and crossover probability of $P_{\text{cx}} = 0.6$.

4.1. Circular shape enforcement, $N = 48$

Geometric characteristics are fundamental in architectural and urban designs. For example, prior research identified four geometric properties to evaluate the quality of urban spaces: smallness, compactness, enclosure, and regularity [45].

In many cultures, the circle is considered the most perfect of all shapes. Circularity or roundness of a shape can be measured in several ways. In this paper, we measure it by maximizing the ratio $A_{\text{tot}}/A_{\text{circ}}$ (see Eq. (3)), where A_{tot} is the total area of the structure projected on the 2D surface (including the enclosed “courtyards”), and A_{circ} is the area of the circle circumscribing that structure. In this way, a dome and a torus are equally and perfectly circular (round). The most well-known architectural example is the *Pantheon* (A.D. 126, Rome, Italy), which models the cosmos and its perfect roundness, representing the sphere of heaven [46]. It is the oldest, continuously used structure in history. In the urban scale, *Palmanova* (A.D. 1636, Udine, Italy) is a concentric city based on a nonagon (roundness = 0.92) , which is a Renaissance example of Utopia — an ideal place where perfection was reflected in the whole of its society.

The purpose of this numerical experiment is to assess whether the proposed algorithm can find the optimal solution for the $N_{\max} = 48$ case with $\delta = 1$, where the objective is to approach a perfectly circular configuration as closely as possible.

[1] The code used to produce the results is available under https://doi.org/10.18150/UB1JDK.

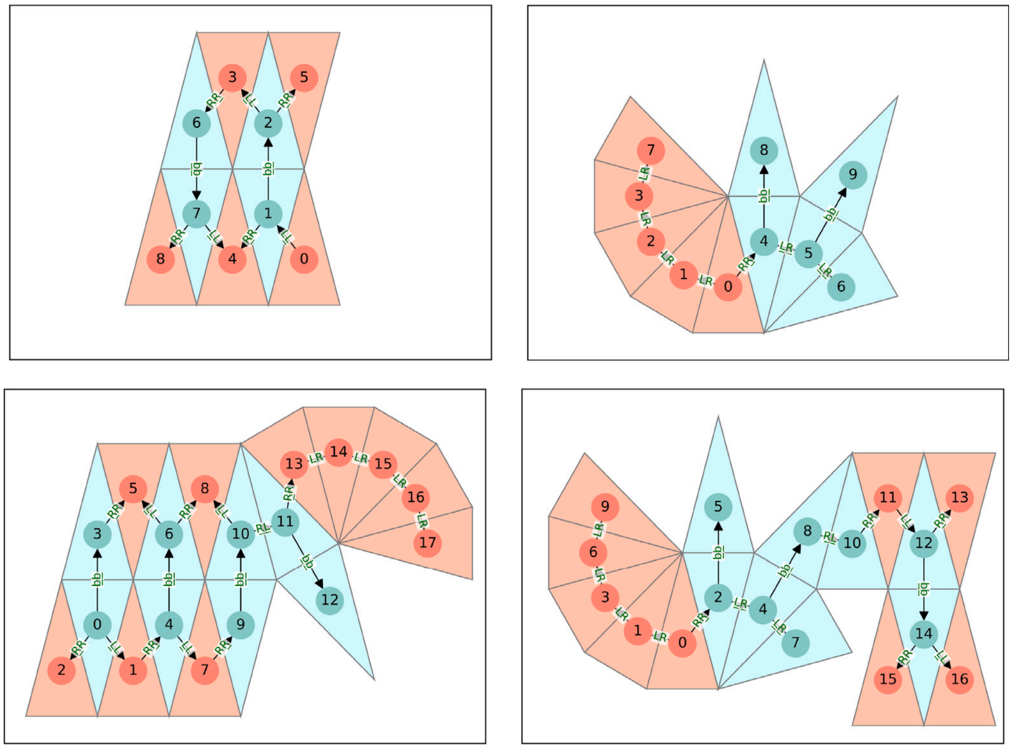


Fig. 22. Top row: two parents: G_1 (left) and G_2 (right). Bottom row: the offspring: $\mathrm{cx}(G_1, G_2)$ (left) and $\mathrm{cx}(G_2, G_1)$ (right). Explanation further in text.

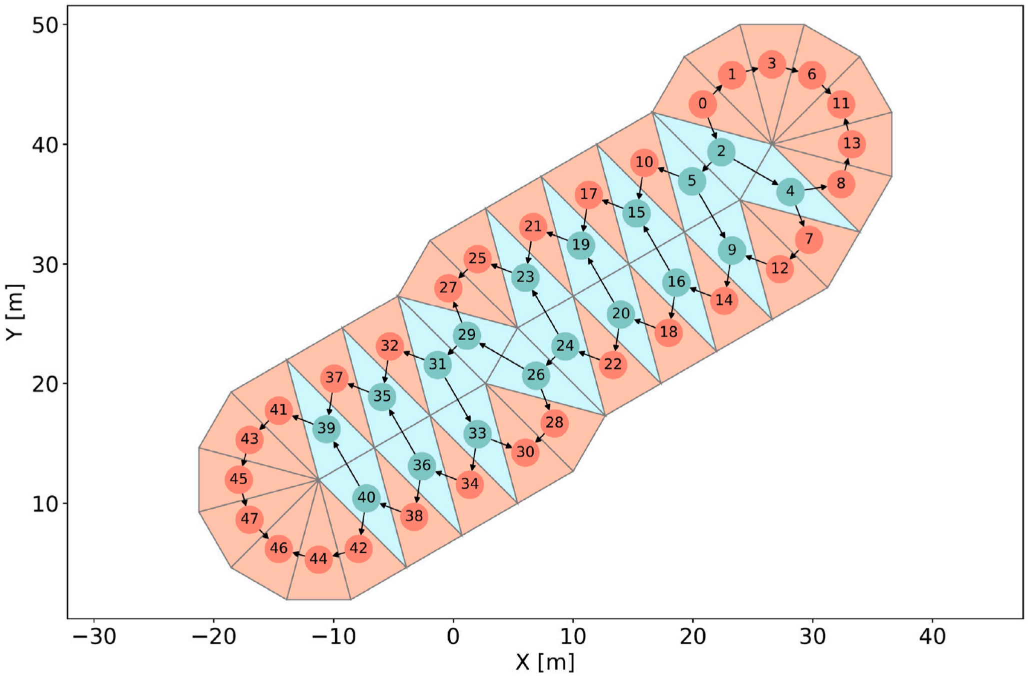


Fig. 23. Example of a straightforward solution maximizing the covered area A_{tot} for $N_{max} = 48$ and $\alpha = \beta = 1, \gamma = \zeta = 0$. This, along with other elongated tunnel-like structures, form a class of simple graphs for $N_{max} \geq 12$ and for $N_{max} \bmod 4 = 0$. The graph and the corresponding triangles of the layout are superimposed (edge types are omitted for clarity).

Intuitively, finding this optimal solution is not difficult; it is feasible to combine modules that form a regular dodecagon (12-gon), as shown in Fig. 24 (bottom, right). Nevertheless, during the evolutionary process, the algorithm must navigate a clear local minimum in the form of a solution that is also a regular 12-gon but consists only of $N = 12$ elements, as depicted in Fig. 24 (top, middle).

Fig. 25 illustrates the progress of the optimization algorithm on 100 trials.

Each trial begins with a random initial population and evolves over 5000 generations. The overall best fitness is plotted (indicating that one of the simulations reached the optimum in just a few steps, specifically at generation = 18 after close inspection of the data), alongside the mean fitness with its standard deviation.

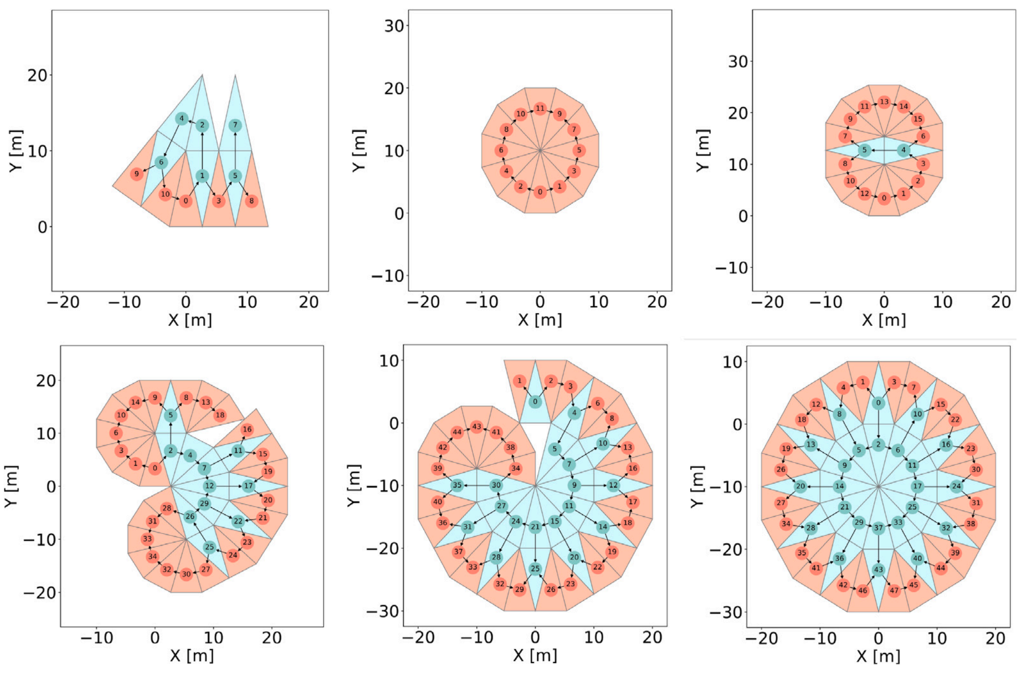


Fig. 24. The best individuals in the population at various stages of the evolutionary process, for generation $T = 1, 2, 3, 6, 252, 260$ (with $f = 1.3, 11.5, 12.9, 19.6, 34.9, 45.8$ respectively). $N_{\max} = 48$, $\alpha = \beta = \gamma = \delta = 1$.

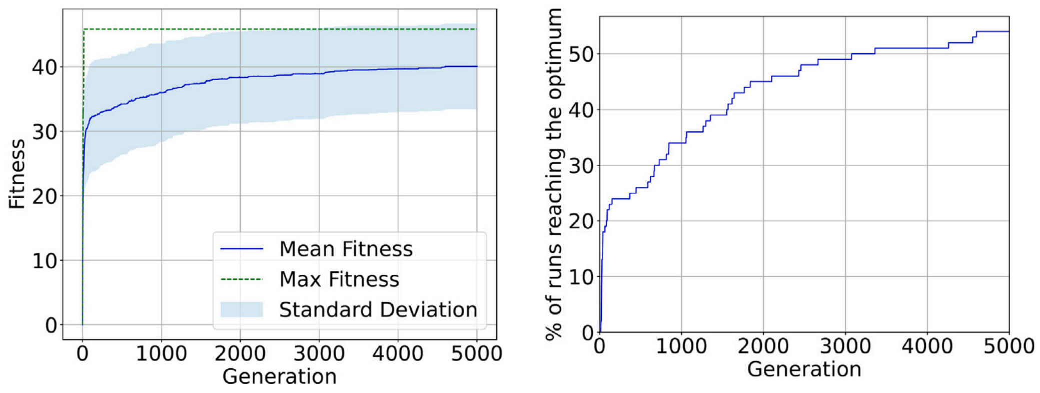


Fig. 25. On the left: the best and mean (with σ) fitness for 100 trials of the evolutionary algorithm as a function of the generation step. On the right: percentage of trials that found the optimal solution at each given generation step.

It can be observed that some trials find the optimal solution very quickly, while others remain in local minima for extended periods. Nearly 20% of the trials reach the optimum within approximately 100 steps, while it takes around 3000 steps for 50% of the trials to achieve the optimum. The evolution of the best individuals during a representative trial is illustrated in Fig. 24. It can be observed that the 12-gon with 12 elements (a local minimum) is found as early as generation = 2 but is replaced by a better-fit individual in the subsequent step. In this case, it took 260 generations to reach the optimum.

4.2. Circular shape enforcement, varied N

The objective of this numerical experiment was to examine a family of solutions resembling a circular shape for varying numbers of available nodes. Figs. 26 and 27 show some of the optimal or near-optimal solutions found for $N_{\max} = 24, \dots, 128$. In some cases, it is advantageous to include a few open sides, allowing the inclusion of extra modules while still approximating a circular shape.

One advantage of using an EA is that it can deliver multiple solutions with similar fitness values that are near-optimal. This enables a choice among functionally equivalent solutions, as seen in the two similar configurations for $N = 64$. Additionally, the algorithm provides insight into expected structural adaptations as $N_{\max}$ changes. For example, in the case of $N_{\max} = 128$, adding only two modules yields a coherent structure.

As shown in Fig. 20, the number of possible solutions increases exponentially with $N_{\max}$. This growth makes it impractical to definitively assess the efficiency of this EA approach, as finding the exact optimum is generally impossible. For special cases, like the solution for $N_{\max} = 48$ presented earlier, optimality can be claimed. For general cases, the algorithm can be executed with various initial conditions to observe the convergence behavior. It is expected that solution quality may deteriorate as $T_{\max}$ and population size remain constant while $N_{\max}$ increases. The structures presented in this section are derived from 20 simulations for each $N_{\max}$, with a maximum number of generations set to $T_{\max} = 5000$ (the calculations were terminated if no improvement occurred over 500 generations).

Figs. 28 and 29 show visualizations of the spatial *Vault-Z* structure — intersecting three horn tori halves, the projection of which is shown in Fig. 27.3.

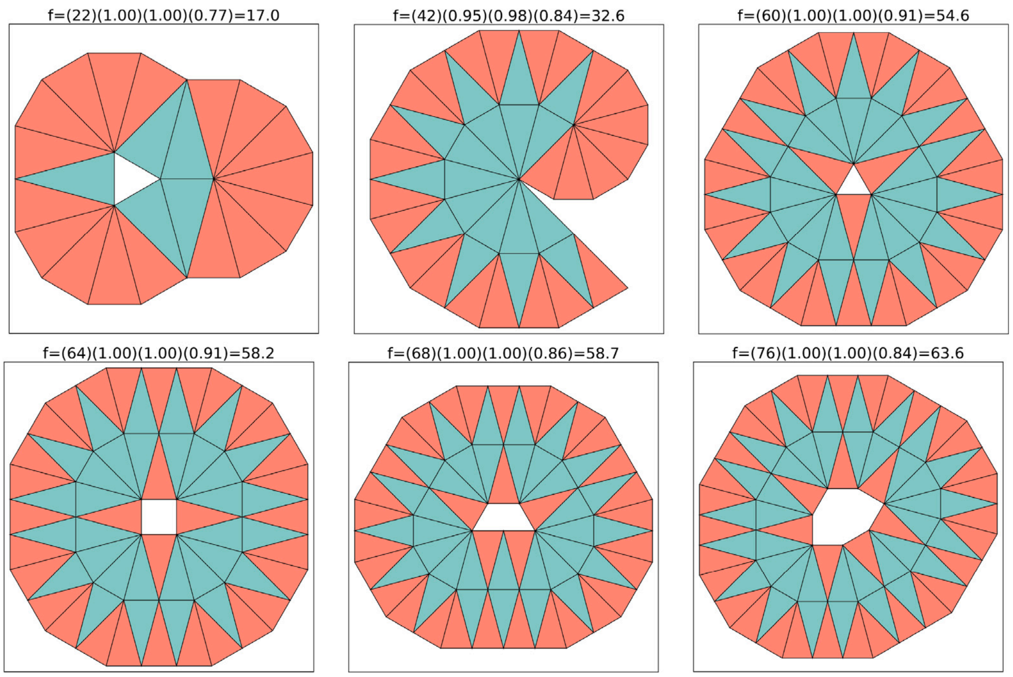

f=(22)(1.00)(1.00)(0.77)=17.0
f=(42)(0.95)(0.98)(0.84)=32.6
f=(60)(1.00)(1.00)(0.91)=54.6
f=(64)(1.00)(1.00)(0.91)=58.2
f=(68)(1.00)(1.00)(0.86)=58.7
f=(76)(1.00)(1.00)(0.84)=63.6

Fig. 26. Top solutions for $N_{\max} = 24, 42, 64, 64, 68, 76$. In the title of each plot, the value of f is presented, along with its multiplication terms so their contribution is visible.

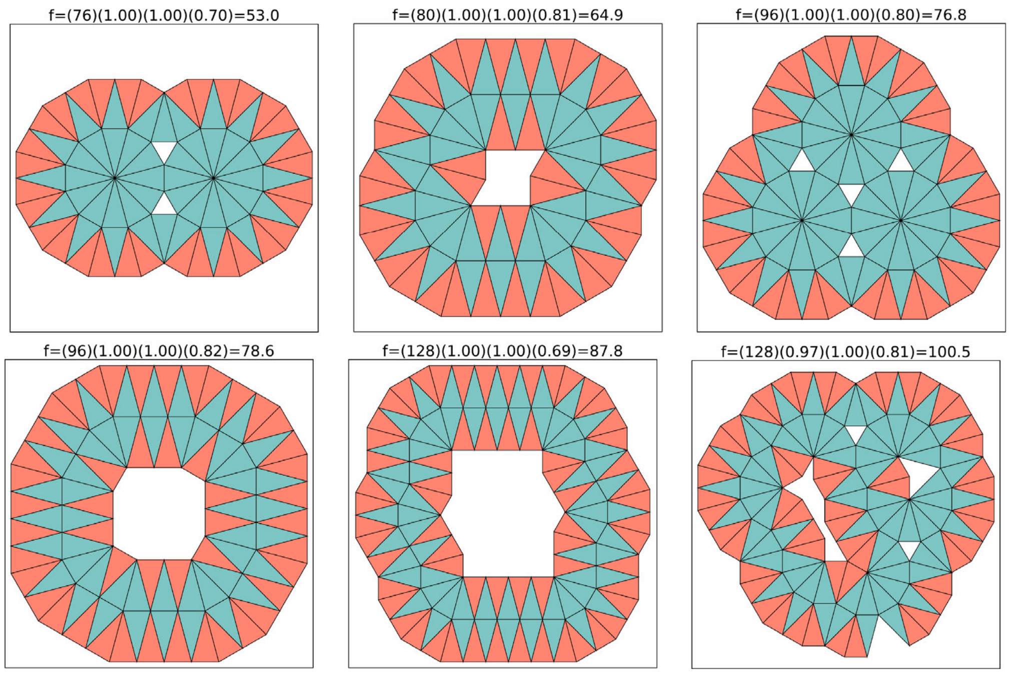

f=(76)(1.00)(1.00)(0.70)=53.0
f=(80)(1.00)(1.00)(0.81)=64.9
f=(96)(1.00)(1.00)(0.80)=76.8
f=(96)(1.00)(1.00)(0.82)=78.6
f=(128)(1.00)(1.00)(0.69)=87.8
f=(128)(0.97)(1.00)(0.81)=100.5

Fig. 27. As in Fig. 26, but for $N_{\max} = 80, 80, 96, 100, 128, 128$. Notice that in the last case, $N_{\max} = 128$, the structure could be closed with only two additional modules.

4.3. Maximization of the enclosed area (courtyard size)

As mentioned above, another way of confining the evolving structures is to maximize the area enclosed by them – in other words, the courtyard size – i.e., the area that cannot be accessed from the outside. At the same time – in the examples shown here – the preference for nodes in cycles and the preference of enclosed structures (minimized number of open sides) are maintained as before.

This approach is motivated by situations where enforcing a circular shape may not be appropriate, while limiting simple linear growth is still desirable. Given this optimality criterion the optimization progresses with growing $N_{\max}$, in three stages:

1. **Low** $N_{\max}$: When $N_{\max}$ is small, the optimization prioritizes connectivity and penalizes disconnected modules and open sides. Under these constraints, the structure tends to grow in a linear or branched form without forming courtyards ($A_{\text{enc}} = 0$).
2. **Intermediate** $N_{\max}$: As $N_{\max}$ increases, the optimization gains more flexibility in allocating modules. At a certain threshold, the benefit of forming an enclosed area (i.e., increasing A_{enc})

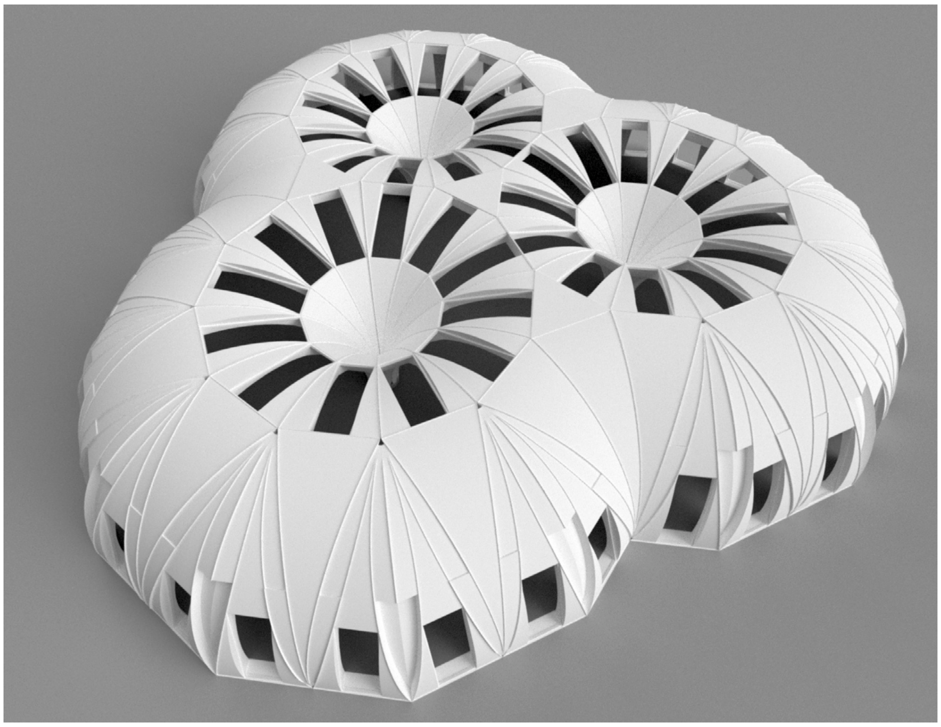


Fig. 28. Bird's-eye view of the $Vault - Z$ triple-torus, whose plan was generated by the algorithm (see Fig. 27.3). The apertures of the $\bigtriangledown$ modules on the perimeter are closed.

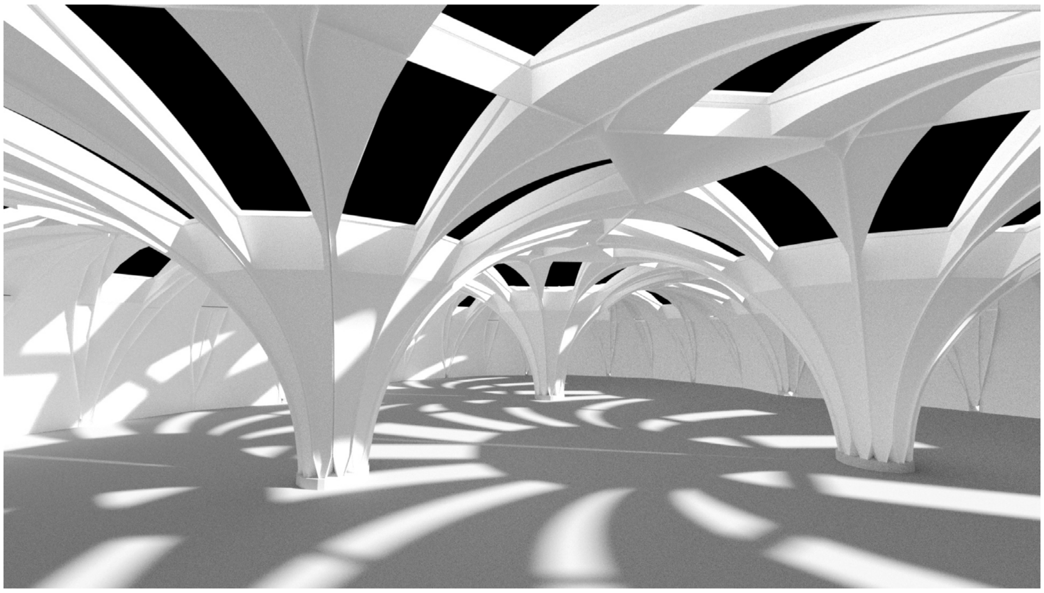


Fig. 29. Interior of the $Vault - Z$ structure, whose plan was generated by the algorithm (shown in Fig. 27.3). Three central "spikes" of the intersecting three half-tori are clearly visible. Apertures of the $\bigtriangleup$ modules on the perimeter are closed.

begins to outweigh the penalty incurred by introducing open sides (N_{open}). Consequently, the structure may start forming partially enclosed regions even if doing so results in some modules having open (unconnected) sides. This trade-off is advantageous because the contribution of A_{enc} to the overall fitness function becomes significant. The configuration at this stage is characterized by the presence of courtyards, although some open sides may occur.

3. **High** N_{max}: When N_{max} becomes sufficiently large, the optimization can achieve both a high A_{enc} and a fully connected boundary. That is, it becomes possible to form a closed chain of modules that completely encloses a region without any open sides. In this final stage, the structure reaches an optimal balance, maximizing size of a courtyard while fully satisfying the connectivity constraints.

Fig. 30 illustrates this process.

The population size (N_{popul}) and the maximum number of evolution cycles (T_{max}) are not increased for larger problems. This limitation is evident in the last case shown in Fig. 30, where $N_{\text{max}} = 256$. This configuration could be further improved to achieve a more circular shape.

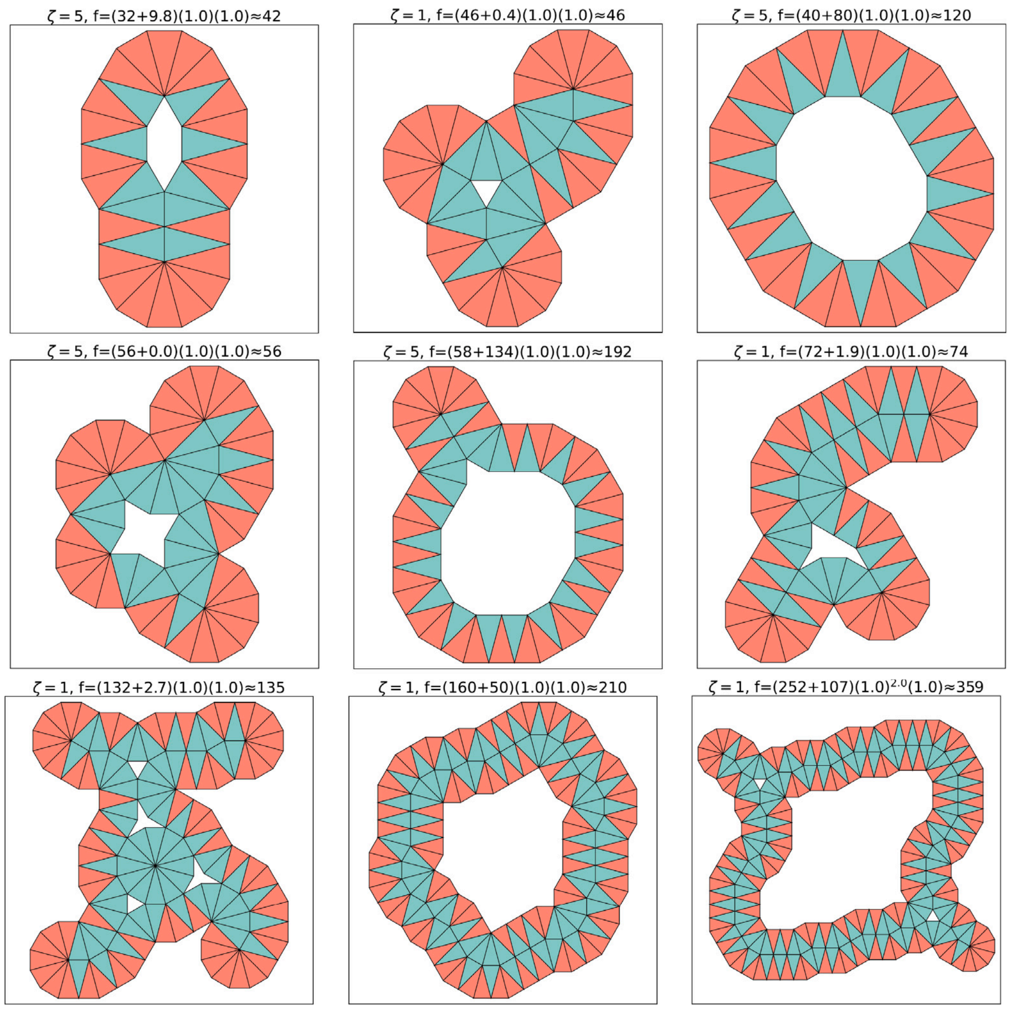

ζ = 5, f=(32+9.8)(1.0)(1.0)≈42
ζ = 1, f=(46+0.4)(1.0)(1.0)≈46
ζ = 5, f=(40+80)(1.0)(1.0)≈120
ζ = 5, f=(56+0.0)(1.0)(1.0)≈56
ζ = 5, f=(58+134)(1.0)(1.0)≈192
ζ = 1, f=(72+1.9)(1.0)(1.0)≈74
ζ = 1, f=(132+2.7)(1.0)(1.0)≈135
ζ = 1, f=(160+50)(1.0)(1.0)≈210
ζ = 1, f=(252+107)(1.0)^2.0(1.0)≈359

Fig. 30. Some of the best solutions from various populations, $N_{\max} = 32, 48, 48, 56, 64, 72, 132, 160, 256$. The enclosed area is maximized by choosing $\zeta = 1$ or $\zeta = 5$ (shown for each respective sub-figure).

This is likely due to the limited number of experiment trials, population size, or generations (which effectively restrict the number of fitness evaluations). However, for this particular case, this limitation is not critical, as with very large $N_{\max}$, the optimal structure is expected to resemble a ring. Nonetheless, this highlights the need for larger population sizes and an increase of $T_{\max}$ to achieve optimal results for very large $N_{\max}$.

4.4. Avoiding obstacles while maximizing the covered area

In this scenario, there are no constraints on the shape, with $\delta = \zeta = 0$; however, the structure is placed in an environment containing obstacles. Intersections with obstacles, and obviously with other modules, are prohibited. Obstacles can take any shape and can be defined as a bitmap with a given pixel resolution. Obstacles represent rocks, hills, or woods.

Embedding the structure in such an environment is done by placing the *anchor module* within the world coordinate system. The positions of the rest of the modules are relative to that module. Each trial starts with a random configuration.

Fig. 31 shows the four best solutions generated during 30 trials with the same geometric constraints. These layouts differ substantially despite similar values of their fitness functions.

This fitness function generates multiple near-optimal solutions. With $\gamma = \zeta = 0$, it is sufficient that all $N_{\max}$ modules are used, the nodes have closed sides, and they form a cycle — provided the environment permits it. In such conditions, the algorithm can generate many equivalent graphs, allowing the architect to select the most suitable configuration.

5. Discussion

The introduced here $Vault - Z$ system suffers from drawbacks typical of any modular system. While it offers clear benefits: prefabrication, scalability, and construction speed, it also imposes strong geometric and topological constraints on the design space. The regularity and connectivity rules required to maintain structural and spatial feasibility may prevent the emergence of certain forms. Furthermore, the dependence on discrete, repeatable modules makes it difficult to accommodate local adaptation to irregular sites or evolving users' needs.

There are several practical challenges related to the production and assembly of physical $Vault - Z$ structures. Moreover, there are many practical issues, such as, thermal properties, robustness, and functionality to be addressed. Completion of the prototype (see Figs. 18 and 19) will give some answers. The primary limitations identified:

- From a common residential building perspective, curved walls are burdened with fundamental disadvantages: problematic functionality and acoustics. However, there are special types of construction, such as emergency shelters, extreme environment outposts, etc., where these two aspects can be sacrificed for advantages like rapid deployability, as well as structural and thermal efficiency.

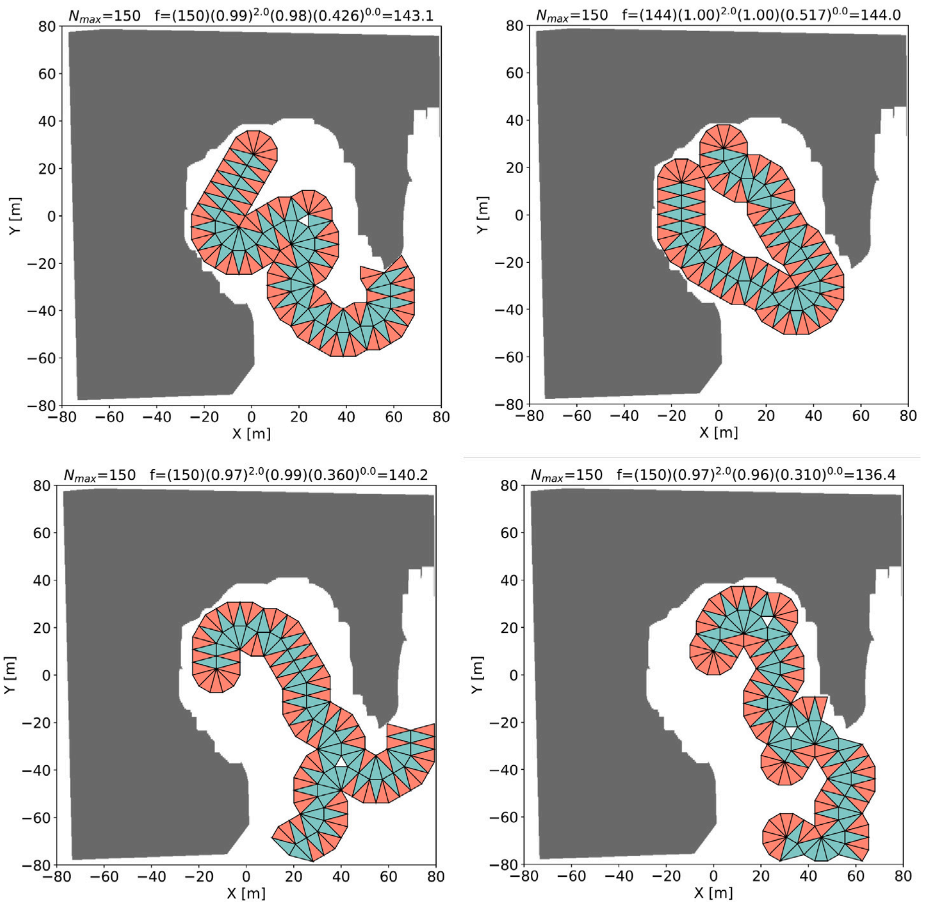


Fig. 31. Collisions with obstacles (shown in gray) are prohibited. Here $\beta = 2$ and $\delta = \zeta = 0$, so there is no dependency on circular shape.

- The modules are relatively large. As photograph Fig. 18.2 indicates, transportation and on-site deployment pose certain challenges, especially if robotic assembly is considered.
- The $Vault-Z$ modules and structures presented seem structurally sound. However, proper structural analysis and topological optimization need to be conducted.
- Presently, the modules have zero tolerance at connection points. For real-life applications, the modules must accommodate manufacturing imperfections and installation inaccuracies on-site.
- Although $Vault-Z$ has much greater potential, at the moment, only single-story structures have been considered. However, even with this simplification, there are several potential applications of the proposed study. Moderate-scale constructions, which require relatively rapid deployment and/or benefit from the possibility of reconfiguration, seem the most appropriate.
- Currently, the algorithm generates a single $Vault-Z$ structure. A generation of several smaller $Vault-Z$s on one site could be more practical in some applications.
- As the number of modules increases, the search space expands so rapidly that finding a globally optimal solution is often impossible. Exceptions to this include trivial solutions, e.g., combining N modules to obtain a structure without geometric constraints on the objective function — such as long, linear tunnels (Fig. 23). Another example – more complex – involves generating connected, closed structures using all available N elements in the presence of obstacles. In such a situation, there can be many different arrangements in a given environment that use all N elements and have no collisions with obstacles. In such a case, the algorithm can be used to generate a number of different, equivalent (in terms of the value of the objective function) solutions, from which the architect will choose the most suitable one.

The presented approach introduces an application of genetic algorithms for evolving graph structures, where each individual population consists of entire graphs represented as nodes and edges in memory, rather than encoded in an adjacency matrix encoding or traditional genotype. The mutation and crossover operators act directly on graph components. This approach retains the structural integrity of the graphs, offering greater flexibility and allowing the evolutionary operators to better exploit graph-specific properties (like cycles and substructures). This has been demonstrated through a series of illustrative examples showing the progression of solutions over the course of evolution. Regarding the optimization procedure, despite its strengths, the proposed framework inherits several known limitations of a graph-topology optimization:

- The evolutionary process explores a combinatorial space of modular graphs, where topological changes – such as adding or removing connections, merging subgraphs – produce non-local, often disruptive effects on design performance. Unlike continuous parametric optimization, there is no smooth interpolation between candidate graphs; the search space is discrete and lacks gradients, which increases the risk of premature convergence or stagnation in local optima (see, e.g., [47]). This occurs often and therefore a relatively large number of trials is required to find a good solution.
- Although the scalar fitness formulation enables flexible prioritization of objectives, it does not explicitly yield Pareto-optimal fronts.
- The fitness evaluation relies on symbolic, domain-specific heuristics (e.g., cycle count, enclosure, circularity) rather than structural or environmental simulation, limiting the approach to conceptual phases.

- The runtime grows quickly with graph size due to repeated graph correctness and collision checks, and the framework – at this point – is implemented in Python without any hardware GPU acceleration.
- The current method is restricted to planar graphs.

6. Conclusions

This paper presented the concept of the innovative $Vault - Z$ modular system for free-form pavilions, its formal representation and an efficient method for creating optimal structures according to given criteria.

Architectural forms enabled by the proposed system are dome- and vault-like pavilions, which are relatively unusual and therefore — interesting, in comparison to the predominantly orthogonal built environment. They are also advantageous in terms of structural performance, particularly in handling compressive stresses [48]. $Vault - Z$ is highly suitable for special types of constructions that can be quickly installed, reconfigured, dismantled, or reused. This approach is particularly useful for post-disaster settlements, and extreme environment outposts. Most importantly, $Vault - Z$ allows for the creation of free-form (extremely) modular pavilions and composed settlements composed of them, which is a unique feature among available modular construction systems. This design flexibility seems particularly suitable for post-disaster settlements. The existing solutions usually reflect the decisions of a restrained group of professionals, rather than the needs, conditions, and cultural context of local inhabitants [49,50]. Moreover, the future production of the proposed system, which is based on a single module is expected to be economical and faster in comparison to traditional pre-fab systems. Simplification, optimization, and automation are the key advantages of the $Vault - Z$ concept.

Directed graphs are used to explicitly define connections between modules. Moreover, by anchoring the first node of the graph embedding to world coordinates it is possible to place the $Vault - Z$ structure within a given environment. As a consequence, various environmental criteria and constraints can be easily incorporated into the fitness function — for instance the amount of earthworks required to set up the structure or avoidance of collisions with existing obstacles. For an example of successful implementation of both criteria, see [51]. Furthermore, since an evolutionary algorithm is used to solve the problem, the fitness function can be easily expanded to accommodate specific requirements set by the designer. Some example solutions for various requirements were presented and visualized. To optimize the solution using an EA, we propose dedicated mutation and crossover operators that act on graphs. The graphs form a population that is evolved to maximize fitness. The result of this optimization, usually over several trials, is a set of candidate solutions with top fitness values. Such configurations can be used directly or serve as a basis for interpretation, further modifications to the parameters of the algorithm, and reiteration of the computation.

Future work will be carried out in three areas: 1. testing and experimenting with the physical prototype; 2. final design of the system for future production; and 3. further development of the optimization algorithms.

1. Presently, the preliminary full-scale prototype is being completed. The physical prototype will allow for full or partial coverage with earth, enabling examination of how such coverage affects thermal insulation, camouflage, and impact resistance. It is also expected that the "organic" forms of $Vault - Z$ will dissipate wind and noise, and impact waves much more effectively than the flat surfaces of traditional buildings. Relevant tests will be carried out.
2. Proper structural analysis and topological optimization of the $Vault - Z$ system is an interesting challenge in the field of structural mechanics. For analogous considerations regarding another Extremely Modular System, see [52]. A successful implementation of structural optimization for the conceptual design of modular bridges, including module topology and spatial orientation optimization was presented in [53]. The $Vault - Z$ module requires further development in order to accommodate the imperfections of the manufacturing and inaccuracies of the installation on-site. Alternative fabrication methods using sustainable technologies and materials (including native materials) will be studied. Another interesting challenge for the near future is the optimal transition between one $Vault - Z$ configuration to another [54].
3. The algorithm will be further developed to address the $Vault - Z$ problem in 3D. It is conceivable to introduce an additional connection type that would allow for the construction of multi-story formations. This, however, introduces a new class of problem, where such configurations will conform to ground elevation maps. For large-scale problems an alternative – hierarchical approach – will be explored. Namely, a set of modular arrangements will be created first (using an evolutionary algorithm). Next, larger structures, potentially disjointed, will be assembled. This approach will be particularly relevant for designing entire settlements comprised of individual modular pavilions with various shapes, sizes, and functions. The combination of Machine Learning and Evolutionary Algorithms is a topical and very interesting approach that will likely lead to significant improvement of the $Vault - Z$ solutions. This novel field of research will be exploited, particularly in the context of 3D structures and larger or hierarchical $Vault - Z$s, where complexity rapidly increases beyond capability of the hitherto developed methods.

CRediT authorship contribution statement

Machi Zawidzki: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Jacek Szklarski:** Writing – original draft, Software, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Z. Soleimani, J.K. Calautit, K. Calautit, B.R. Hughes, Climatic analysis of ventilation and thermal performance of a dome building with roof vent, Eng. Sustain. 171 (2018) 411–424, http://dx.doi.org/10.1680/jensu.16.00018.

[2] Andrew J. Marsh, Domes and solar radiation, Nat. Freq. (002) (2006) URL https://web.archive.org/web/20180829020236/http://naturalfrequency.com/articles/solardomes.

[3] Tahsin Başaran, Thermal analysis of the domed vernacular houses of Harran, Turkey, Indoor Built Environ. 20 (5) (2011) 543–554, http://dx.doi.org/10.1177/1420326X11411237.

[4] Nicos Kalapodis, Georgios Kampas, Olga-Joan Ktenidou, A review towards the design of extraterrestrial structures: From regolith to human outposts, Acta Astronaut. 175 (2020) 540–569, http://dx.doi.org/10.1016/j.actaastro.2020.05.038.

[5] Satoshi Inoue, Kiyoshi Sugino, Masahiro Katou, Hiroyuki Imaizumi, Speech transmission performance and the effect of acoustical remedies in a dome, Appl. Acoust. 70 (1) (2009) 221–230, http://dx.doi.org/10.1016/j.apacoust.2007.12.004.

[6] Mostafa Refat Ismail, Hazem Eldaly, Acoustic of monolithic dome structures, Front. Archit. Res. 7 (1) (2018) 56–66, http://dx.doi.org/10.1016/j.foar.2017.11.002.

[7] Google Maps - street view, Ustrzycka Street, Warsaw, Poland, 2022, URL https://maps.app.goo.gl/3kYUZgtL5kJjBGXs8.

[8] Google Maps, Bowolningen, Den Bosch, the Netherlands, 2025, URL https://maps.app.goo.gl/8QUZ6EFFVtP2TH4M9.

[9] Google Maps - street view, Bowolningen, Den Bosch, the Netherlands, 2023, URL https://maps.app.goo.gl/vc1PXkv14hpTSZty9.

[10] Monolithic Dome Institute, Home floor plans, 2024, URL https://monolithicdome.com/floor-plans.

[11] M. Luz Castro Pena, Adrián Carballal, Nereida Rodríguez-Fernández, Iria Santos, Juan Romero, Artificial intelligence applied to conceptual design. A review of its use in architecture, Autom. Constr. 124 (2021) 103550, http://dx.doi.org/10.1016/j.autcon.2021.103550.

[12] Samuel S.Y. Wong, Keith C.C. Chan, EvoArch: An evolutionary algorithm for architectural layout design, Comput.-Aided Des. 41 (9) (2009) 649–667, http://dx.doi.org/10.1016/j.cad.2009.04.005.

[13] Ralph Evins, A review of computational optimisation methods applied to sustainable building design, Renew. Sustain. Energy Rev. 22 (2013) 230–245, http://dx.doi.org/10.1016/j.rser.2013.02.004.

[14] Yaochu Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Comput. 9 (1) (2005) 3–12, http://dx.doi.org/10.1007/s00500-003-0328-5.

[15] Boris A. Trakhtenbrot, A survey of Russian approaches to perebor (brute-force searches) algorithms, Ann. Hist. Comput. 6 (4) (2008) 384–400, http://dx.doi.org/10.1109/MAHC.1984.10036.

[16] C. Robert Taylor, Dynamic programming and the curses of dimensionality, in: Applications of Dynamic Programming to Agricultural Decision Problems, CRC Press, 2019, pp. 1–10, URL https://www.taylorfrancis.com/chapters/edit/10.1201/9780429040917-1/dynamic-programming-curses-dimensionality-robert-taylor.

[17] Bochra Rabbouch, Hana Rabbouch, Foued Saâdaoui, Rafaa Mraihi, Foundations of combinatorial optimization, heuristics, and metaheuristics, in: Comprehensive Metaheuristics, Elsevier, 2023, pp. 407–438, http://dx.doi.org/10.1016/B978-0-323-91781-0.00022-3.

[18] Tommy R. Jensen, Bjarne Toft, Graph Coloring Problems, John Wiley & Sons, 2011, http://dx.doi.org/10.1002/9781118032497.

[19] Paolo Toth, Daniele Vigo, The Vehicle Routing Problem, SIAM, 2002, http://dx.doi.org/10.1137/1.9780898718515.

[20] Kris Braekers, Katrien Ramaekers, Inneke Van Nieuwenhuyse, The vehicle routing problem: State of the art classification and review, Comput. Ind. Eng. 99 (2016) 300–313, http://dx.doi.org/10.1016/j.cie.2015.12.007.

[21] Gilbert Laporte, The traveling salesman problem: An overview of exact and approximate algorithms, European J. Oper. Res. 59 (2) (1992) 231–247, http://dx.doi.org/10.1016/0377-2217(92)90138-Y.

[22] Gregory Gutin, Abraham P. Punnen, The Traveling Salesman Problem and Its Variations, vol. 12, Springer Science & Business Media, 2006, http://dx.doi.org/10.1007/b101971.

[23] Lance Fortnow, The status of the P versus NP problem, Commun. ACM 52 (9) (2009) 78–86, http://dx.doi.org/10.1145/1562164.1562186.

[24] Vincent A. Cicirello, Cycle mutation: Evolving permutations via cycle induction, Appl. Sci. (ISSN: 2076-3417) 12 (11) (2022) http://dx.doi.org/10.3390/app12115506.

[25] El-Ghazali Talbi, Metaheuristics: From Design to Implementation, John Wiley & Sons, 2009, http://dx.doi.org/10.1002/9780470496916.

[26] Thomas Bartz-Beielstein, Jürgen Branke, Jörn Mehnen, Olaf Mersmann, Evolutionary algorithms, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 4 (3) (2014) 178–195, http://dx.doi.org/10.1002/widm.1124.

[27] Adam Slowik, Halina Kwasnicka, Evolutionary algorithms and their applications to engineering problems, Neural Comput. Appl. 32 (2020) 12363–12379, http://dx.doi.org/10.1007/s00521-020-04832-8.

[28] Carlos A. Coello Coello, Evolutionary Algorithms for Solving Multi-Objective Problems, Springer, 2007, http://dx.doi.org/10.1007/978-0-387-36797-2.

[29] Machi Zawidzki, Marcin Bator, Application of evolutionary algorithm for optimization of the sequence of initial conditions for the cellular automaton-based shading, J. Cell. Autom. 7 (2012) URL https://www.oldcitypublishing.com/journals/jca-home/jca-issue-contents/jca-volume-7-number-5-6-2012/jca-7-5-6-p-363-384/.

[30] Machi Zawidzki, Application of evolutionary algorithms for optimum layout of Truss-Z linkage in an environment with obstacles, Adv. Eng. Softw. 65 (2013) 43–59, http://dx.doi.org/10.1016/j.advengsoft.2013.04.022.

[31] Machi Zawidzki, Optimization of multi-branch Truss-Z based on evolution strategy, Adv. Eng. Softw. 100 (2016) 113–125, http://dx.doi.org/10.1016/j.advengsoft.2016.07.015.

[32] Machi Zawidzki, Implementing cellular automata for dynamically shading a building facade, Complex Systems 18 (3) (2009) 287, http://dx.doi.org/10.25088/ComplexSystems.18.3.287.

[33] Machi Zawidzki, Ichiro Fujieda, The prototyping of a shading device controlled by a cellular automaton, Complex Systems 19 (2) (2010) 157, http://dx.doi.org/10.25088/ComplexSystems.19.2.157.

[34] Yanan Wang, Yan Pei, A comprehensive survey on interactive evolutionary computation in the first two decades of the 21st century, Appl. Soft Comput. 164 (2024) 111950, http://dx.doi.org/10.1016/j.asoc.2024.111950.

[35] Severi Uusitalo, Anna Kantosalo, Antti Salovaara, Tapio Takala, Christian Guckelsberger, Creative collaboration with interactive evolutionary algorithms: a reflective exploratory design study, Genet. Program. Evol. Mach. 25 (1) (2024) 4, http://dx.doi.org/10.1007/s10710-023-09477-9.

[36] Franz Rothlauf, Representations for Genetic and Evolutionary Algorithms, Springer, 2006, http://dx.doi.org/10.1007/3-540-32444-5, ISBN 978-3-540-32444-5.

[37] Agoston E. Eiben, James E. Smith, Introduction to Evolutionary Computing, Springer, 2015, http://dx.doi.org/10.1007/978-3-662-44874-8.

[38] Kenneth Mark Bryden, Daniel A. Ashlock, Steven Corns, Stephen J. Willson, Graph-based evolutionary algorithms, IEEE Trans. Evol. Comput. 10 (5) (2006) 550–567, http://dx.doi.org/10.1109/TEVC.2005.863128.

[39] Hannes Kneiding, David Balcells, Augmenting genetic algorithms with machine learning for inverse molecular design, Chem. Sci. (ISSN: 2041-6520) 15 (38) (2024) 15522–15539, http://dx.doi.org/10.1039/d4sc02934h, URL https://www.sciencedirect.com/science/article/pii/S2041652024013695.

[40] Jan H. Jensen, A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space, Chem. Sci. 10 (12) (2019) 3567–3572, http://dx.doi.org/10.1145/3638530.3654349.

[41] Léo Françoso D.P. Sotto, Paul Kaufmann, Timothy Atkinson, Roman Kalkreuth, Márcio Porto Basgalupp, A study on graph representations for genetic programming, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, 2020, pp. 931–939, http://dx.doi.org/10.1145/3377930.3390234.

[42] Jared Murphy, Devroop Kar, Joshua Karns, Travis Desell, EXA-GP: Unifying graph-based genetic programming and neuroevolution for explainable time series forecasting, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2024, pp. 523–526, http://dx.doi.org/10.1145/3638530.3654349.

[43] Kaichen Ouyang, Shengwei Fu, Zong Ke, Renxiang Guan, Ke Liang, Dayu Hu, Learn from global correlations: Enhancing evolutionary algorithm via spectral GNN, 2025, URL https://arxiv.org/abs/2412.17629.

[44] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, Christian Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. 13 (1) (2012) 2171–2175, http://dx.doi.org/10.5555/2503308.2503311.

[45] Machi Zawidzki, Automated geometrical evaluation of a plaza (town square), Adv. Eng. Softw. 96 (2016) 58–69, http://dx.doi.org/10.1016/j.advengsoft.2014.11.004.

[46] Kim Williams, The well-rounded architect: Spheres and circles in architectural design, in: Reza Sarhangi (Ed.), Bridges: Mathematical Connections in Art, Music, and Science, Bridges Conference, Southwestern College, Winfield, Kansas, ISBN: 0-9665201-1-4, 1999, pp. 173–184, URL http://archive.bridgesmathart.org/1999/bridges1999-173.html.

[47] Mathias Giger, Paolo Ermanni, Evolutionary truss topology optimization using a graph-based parameterization concept, Struct. Multidiscip. Optim. 32 (4) (2006) 313–326, http://dx.doi.org/10.1007/s00158-006-0028-8.

[48] Moayyad Al-Nasra, Finite element investigation of dome-like structures, ARPN J. Eng. Appl. Sci. 12 (11) (2017) 3445–3450, URL https://aurak.ac.ae/publications/Finite-Element-Investigation-of-Dome-Like-Structures.pdf.

[49] D. Alexander, Post-disaster reconstruction: meeting stakeholder interests: proceedings of a conference held at the Scuola di sanità militare, Florence, Italy, 17-19 May 2006, Post-Disaster Reconstr. (2007) 0, http://dx.doi.org/10.36253/978-88-8453-611-2.

[50] S. El-Masri, Post-war reconstruction. Participatory approaches to rebuilding the damaged villages of Lebanon: a case study of al-Burjain, Habitat Int. 25 (4) (2001) 535–557, http://dx.doi.org/10.1016/S0197-3975(01)00023-6.

[51] Machi Zawidzki, Jacek Szklarski, Effective multi-objective discrete optimization of Truss -Z layouts using a GPU, Appl. Soft Comput. 70 (2018) 501–512, http://dx.doi.org/10.1016/j.asoc.2018.05.042.

[52] Machi Zawidzki, Łukasz Jankowski, Optimization of modular Truss-Z by minimum-mass design under equivalent stress constraint, Smart Struct. Syst. 21 (6) (2018) 715–725, http://dx.doi.org/10.12989/sss.2018.21.6.715.

[53] Alexis Tugilimana, Ashley P. Thrall, Rajan Filomeno Coelho, Conceptual design of modular bridges including layout optimization and component reusability, J. Bridg. Eng. 22 (11) (2017) 04017094, http://dx.doi.org/10.1061/(asce)be.1943-5592.0001138.

[54] Machi Zawidzki, Jacek Szklarski, Transformations of Arm-Z modular manipulator with particle swarm optimization, Adv. Eng. Softw. 126 (2018) 147–160, http://dx.doi.org/10.1016/j.advengsoft.2018.05.003.